

Table of Contents

Introduction

1 Elephant Robotics	1.1
1 Company Introduction	1.1.1
2 Development History	1.1.2
3 How to Read	1.1.3
2 mechArm 270-PI	2.1
1 Introduction to Robot Parameters	2.1.1
2 Robotic Arm Electrical Interface	2.1.2
3 Unpackaging and First-time Use	2.1.3
4 Development Environment and Construction	2.1.4
3 FAQ	2.2
1 How to Ask Questions Gracefully	2.2.1
2 Driver-related	2.2.2
3 Software	2.2.3
4 Hardware	2.2.4

Preparations

4 Background Knowledge	3.1
4.1 Robot Arm	3.1.1
4.2 Electronic Background Knowledge	3.1.2
4.3 Knowledge of Motor and Servo	3.1.3
4.4 Robot Kinematics	3.1.4
1 myCobot Series 6-Axis Collaborative Robotic Arm	3.1.4.1
5 Basic Usage	3.2
Safety_Instruction	3.2.1
5.1 myStudio	3.2.2
1 Environment Building	3.2.2.1
2 Burning and Updating Firmwares	3.2.2.2
5.2 Factory Firmware Introduction	3.2.3
5.2.1 Drag Teaching	3.2.3.1
Microprocessor Class	3.2.3.1.1
5.2.2 Zero Point Calibration	3.2.3.2
Microprocessor Class	3.2.3.2.1
5.2.3 Communication Forwarding	3.2.3.3
Microprocessor Class	3.2.3.3.1
5.2.4 Connection Detection	3.2.3.4

Microprocessor Class	3.2.3.4.1
5.3 First-time Use	3.2.4
5.4 PI Robot Instructions	3.2.5
1 Introduction to PI version Robot	3.2.5.1
2 Description of Basic System Functions	3.2.5.2
3 Description of Common Hardware Interfaces	3.2.5.3
5.5 Downloads	3.2.6

Development and API

6 myBlockly	4.1
1 The First-Time Use	4.1.1
2 Setting the Color of RGB Light Panel	4.1.2
3 Setting All Arms to Starting Point	4.1.3
4 Controlling Single-Joint Motion	4.1.4
5 Controlling Multi-Joint Motion	4.1.5
6 Swinging Arms Left and Right	4.1.6
7 Let Robot Dance	4.1.7
8 The Use of Gripper	4.1.8
9 The Use of Sucking Pump	4.1.9
10 Gripper Test	4.1.10
11 IO Test	4.1.11
7 Python	4.2
1 Environment Building	4.2.1
2 Introduction to API	4.2.2
3 Joint Control	4.2.3
4 Coordinate Control	4.2.4
5 IO Control	4.2.5
6 Gripper Control	4.2.6
7 TCP/IP Control	4.2.7
8 Drag to teach	4.2.8
9 Videos and Codes for Display	4.2.9
8 ROS	4.3
8.1 ROS1	4.3.1
8.1.1 Environment Building	4.3.1.1
8.1.2 ROS Basics	4.3.1.2
8.1.3 Introduction and Use Of Rviz	4.3.1.3
mechArm_270	4.3.1.3.1
8.1.4 Moveit	4.3.1.4
mechArm_270	4.3.1.4.1

8.2 ROS 2	4.3.2
8.2.1 Installation of ROS2	4.3.2.1
8.2.2 Basic Tutorial	4.3.2.2
8.2.3 Rviz Introduction and Use	4.3.2.3
mechArm 270	4.3.2.3.1
9 Communication and Message Commands	4.4
10 Comment & Feedback	4.5

myCobot: From 0 to 1



1.1 Why do we design myCobot

An entry-level collaborative robot arm that everyone can learn and play

The original design of myCobot is to help friends who are interested in 6-axis series robot to learn it from entry to master, creating unprecedented experience and teaching value.

What you can learn

Robotics is based on rigid body kinematics and dynamics, but also an interdisciplinary subject that combines hardware, software, algorithm and control.

With myCobot, you can learn that

- **Hardware**
 - Embedded Microcontroller Based on ESP32
 - Motor and Steering Gear
 - M5Stack Basic/ Atom
- **Software**
 - Arduino开发环境
 - C++

- Python
- ROS, MoveIt
- Communication Data
- Virtual Machines & Linux (visual system)
- Algorithm
 - Series Manipulator
 - Coordinate and Coordinate Transformation坐标与坐标转换
 - DH Parameters
 - Kinematics
 - Manipulator Algorithm (e.g. dynamics)
- Machine Vision (Vision Set)
 - Color Recognition
 - Image Recognition
 - Hand-Eye Calibration
 - See and Grab
- Extended Applications
 - End-effector: gripper, suction pump, etc.
 - Robot Suit & Industry 4.0 Applications



Parts of Gitbook

View the directory on the left to jump

There are four major parts of Gitbook :

- **Introduction & Quick Start**
 - **Introduction** -- introduce what myCobot is and its main features, etc.
 - **How to Read** -- help you read Gitbook efficiently according to your learning level and knowledge background

- **Use Cases** -- you can know exactly what use cases you can accomplish with
 - **Quick Start** -- learn the unboxing of your myCobot, and its first boot and use
-

- **Preparation before Development**

- **Background Knowledge** -- learn about tools, industrial robots, algorithms, software, hardware, etc.
- **Hardware Learning** -- learn about embedded hardware, structural components, electronic components, etc.
- **Purpose of Use** -- identify the purpose you want to use it for, and complete the study related to your task

- **Development and Use**

- **Development Environment** -- learn to use Arduino, ROS, uiFlow, roboFlow, python and others development environment to develop myCobot
- **Accessories** -- learn to use myCobot with different accessories, such as bases, grippers, suction pumps and so on
- **Machine Vision** -- learn to control myCobot under the guidance of machine vision
- **Robot Modification** -- learn how to modify myCobot into a 4 or 5 axis manipulator

- **myCobot Suit**

- **Intelligent Warehouse**: learn how to use myCobot to carry different objects
- **Artificial Intelligence**: learn how to control myCobot to grasp objects intelligently under the guidance of machine vision
- **Industry 4.0**: learn how to grasp and place objects intelligently by simulating production line

Information Source

- **Official website:** www.elephantrobotics.com
 - **Tutorial video:** <https://www.youtube.com/channel/UC68l2RaRF2Mp8fzpCTzNBfA>
 - **Shop website:** <https://shop.elephantrobotics.com>
 - [Download PDF](#)
-

Contact Us

If you have any other questions, you can contact us as follows.

We will answer you as soon as possible (working day 9:30-18:30)

- Twitter: myCobot Official\@CobotMy
 - Facebook: <https://www.facebook.com/MyCobot-116558893805177>
 - Mail: support@elephantrobotics.com
-

Elephant Robotics



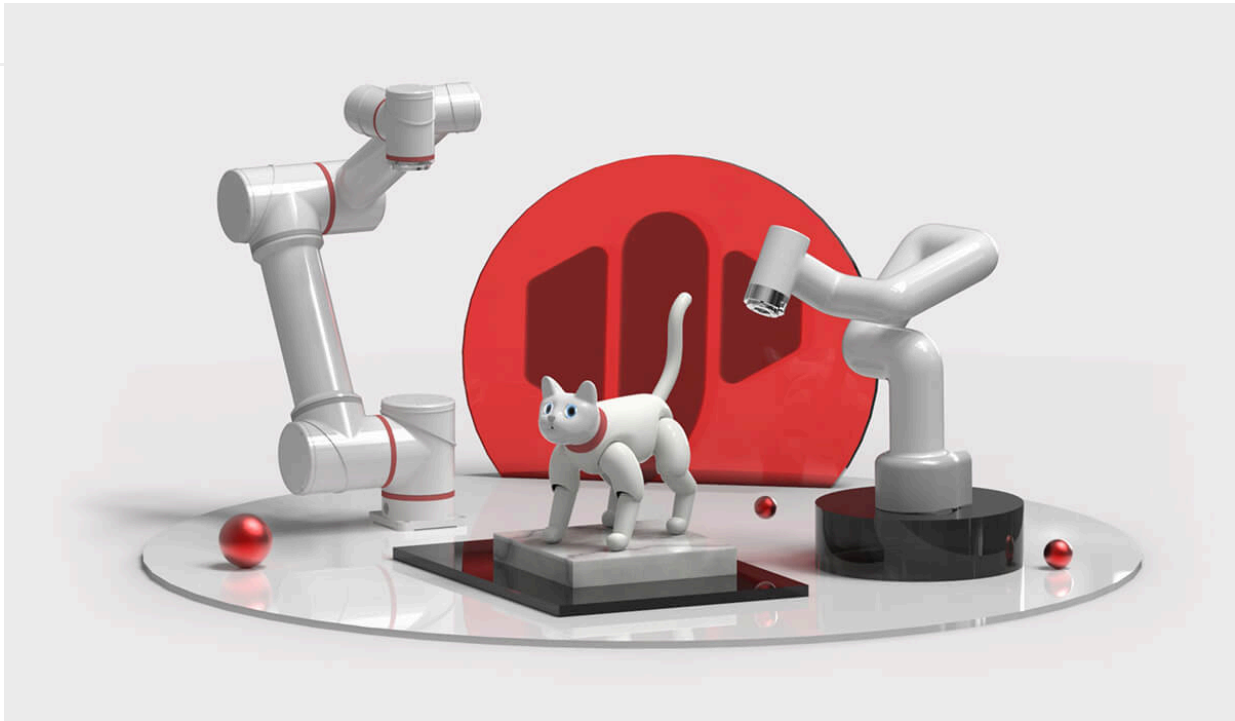
1 Company Introduction

Elephant Robotics is based in Shenzhen, China, a high-tech company which focuses on the design, research and development of robots and automation solutions.

We are devoted to providing highly flexible robots, easy-to-learn operating systems and intelligent automation solutions for robot education, scientific research institutions, business situations and industrial production. Our product quality and intelligent solutions have obtained unanimous acceptance and favorable comments from a number of world top 500 enterprises & factories in South Korea, Japan, America, Germany, Italy, Greece, etc.

Abiding by the vision of "enjoy robots world, Elephant Robotics initiates collaborative work between human and robots to make robots be good life and work helpers for human so as to free people from simple, repeated and dull jobs and give full play to the advantages of human-robot coordination, thus improving work efficiency and helping human to create a nice, new life.

In the future, Elephant Robotics hopes to promote the development of the robot industry through a new generation of cutting-edge technology, and starts a new era of automation and intelligence with its customers hand in hand.



2 Development History

In August 2016, Elephant Robotics was established.

In August 2016, entered HAX Incubator and obtained SOSV seed round investment.

In July 2017, the two founders were included in Forbes Asia's "30 Business Elites under Age 30".

In October 2017, published the fifth generation of single-arm industrial cobot called Elephant S.

In April 2018, obtained angel round investment from Cloud Angel Fund.

In June 2018, was awarded "Intelligent Manufacture Entrepreneurship MBA Award" by CKGSB.

In June 2018, was awarded "Startup Accelerator X-elerator Award" operated by Tsinghua University.

In November 2018, won the second place in the Asian Smart Hardware Competition in Shenzhen Division

In November 2018, obtained the "Most Invested Company Award" in GaogongGold Globe Award.

In March 2019, obtained the "Leading Person Award" in Gaogong Gold Globe Award.

In April 2019, obtained Catbot "Industrial Robot Innovation Award".

In September 2019, attended Huawei European Eco-Conference (HCE) and became a member of Huawei eco-partners.

In November 2019, Elephant Robotics attended the IROS International Conference on Intelligent Robots and Systems jointly with Harbin Institute of Technology.

In December 2019, obtained "Gaogong 2019 Innovation Technology Award".

In December 2019, was awarded as one of the Gaogong 2019 Top 10 Fast Growing Enterprises.

In December 2019, was awarded the "Emerging Enterprise Award" in the industrial robotics segment field of Shenzhen equipment industry.

In December 2019, launched the first type of bionic robotic cat called MarsCat in the world.

In May 2020, the founders obtained \"Shenzhen Robot Emerging Talent Award\" in 2019.

In October 2020, launched the smallest six-axis cobot named myCobot in the world.

In March 2020, launched the smallest cobot named myCobotPro 320 for scientific research in the world.

In May 2021, the Mars bionic cat named MarsCat was reported by several media such as Xinhua Finance, China Daily, Nanjing Daily, Harbin Daily, etc.

In July 2021, published the seat for the smallest hybrid robot, a baby elephant moving robot called myAGV.

In September 2021, launched the world's first type of fully wrapped four-axis robot arm, a tiny elephant palletizing robot arm called myPalletizer.

3 Related Links

- Official website: <https://www.elephantrobotics.com>
- Purchase link
 - shopify: <https://shop.elephantrobotics.com/>
- Video
 - bilibili: <https://space.bilibili.com/2126215657>
 - youtube: <https://www.youtube.com/c/Elephantrobotics>

4 Contact Us

If you have any other problems, contact us via the ways below. + Email: We will give a reply within 1-2 business days; **Email + WeChat**: We provide one-to-one service only for those users who have purchased myCobot via WeChat.

Development history of my series of products

Development History

In October 2020, we launched the smallest six-axis cobot named **myCobot 280-M5** in the world.

In December 2020, **myCobot 280** was put on the market.

In April 2021, **myCobot 320** products was put on the market.

In May 2021, we published the Raspberry Pi version of **myCobot 280**.

In June 2021, we published the Raspberry Pi version of **myCobot 320**.

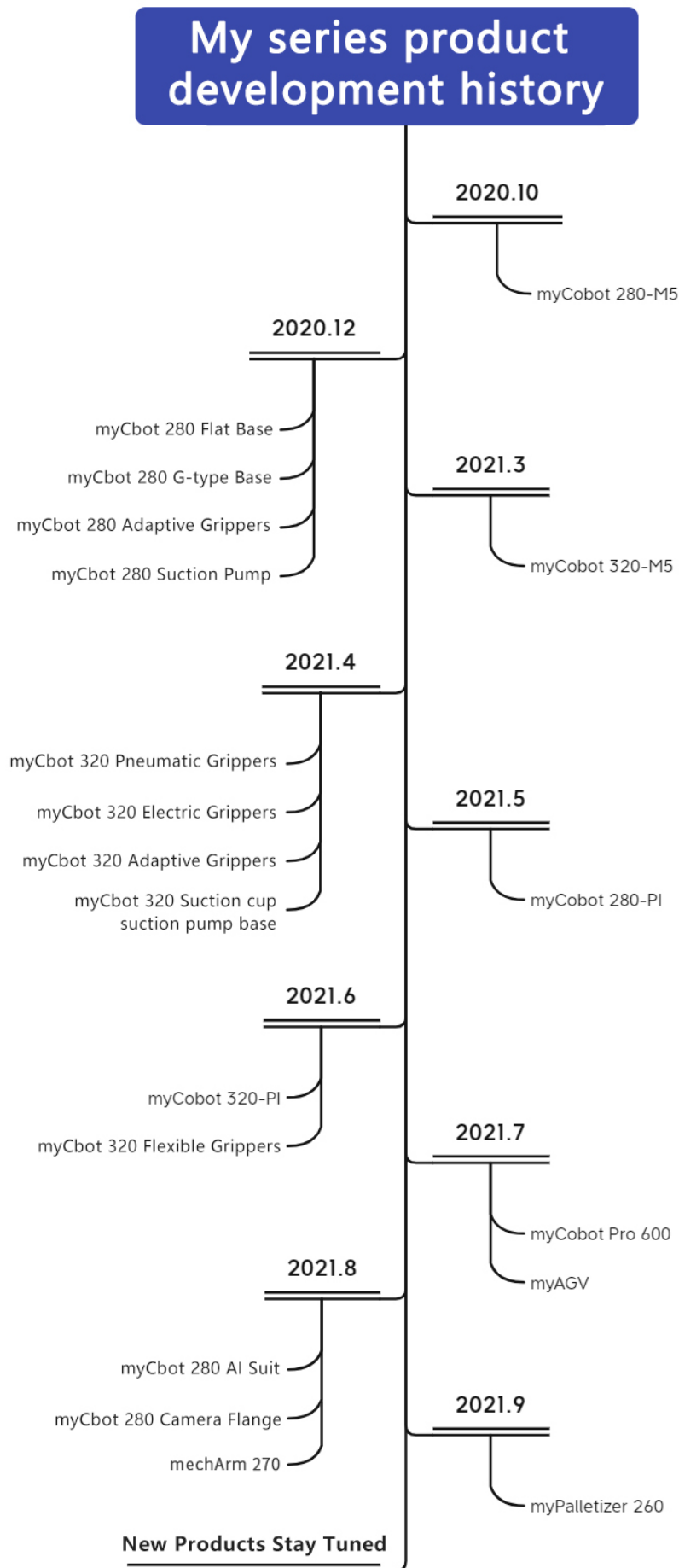
In July 2021, we published the commercial version of a babyelephant coordinative robot arm called **myCobot Pro 600**.

In July 2021, we published the smallest hybrid robot, a baby elephant moving robot called **myAGV**.

In August 2021, **AI kits** and the computer vision enabled for visual machines were put on the market.

In August 2021, The world's most compact and portable small six-axis mechanical **mechArm 270** series came out

In September 2021, launched the world's first type of fully wrapped four-axis robot arm, a tiny palletizing robot arm called **myPalletizer**.



How to read



1 The target of reading the book

The book is designed to help you reach the following targets.

Main targets

- Understand the basic use of mechanisms, electronics and software related to robot arms.
- Understand the basic principle, joints, coordinates, terms, control and other information of the robot arm. Able to do simple forward and inverse kinematics calculations.
- Understand the basic operations of API-controlled robot arms and the robot arms controlled by myBlockly graphical programming language.

Extended targets

- Understand the image recognition algorithms related to machine vision.
- Understand the building of robot vision scenes and the methods and strategies of coordination between vision and robot arm.
- Grasp all skills of artificial intelligence (AI) packages.

2 Division of your background

You should read the book based on your background. We divide the background and related knowledge into three degrees:

Degree	Background	Skills	Estimated time of learning	Recommended platform
Beginner	Specialty related to information, electronics and automation	Understand a kind of programming language and the basic knowledge related to electronics	100 hours	myBlockly
Advanced	Understand Arduino or similar hardwares, servo and programming, IO interface, etc	Able to debug API and interfaces, and understand communication	50 hours	Arduino
Professional	The readers used at least one industrial or consumer robot arms, and have the ability to develop hardware and software	Understand the Cartesian coordinate system, joint control, and basic use of robots	30 hours	Random

3 Learning Steps and Time

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
1	Quick unpacking	1 Drag teaching	1. The accessories for myCobot 2. drive the robot to perform drag teaching	1 hour
2	Background knowledge learning	1. Use background of industrial robots; 2. coordinate and space learning, and Cartesian 3D coordinate and rotation, xyz; 3. joint and coordinate control of industrial robots	1. Joint control and recurrence of robot joints 2. speed control of robots 3. control and cycle of robot coordinate points	5 hours
3	Hardware learning	1. Principles and operations of embedded electronics 2. the principle and knowledge of servo and motor 3. actuator learning	1. Basic/atom control and driving 2. driving and motion of servo 3. robot accessory learning	5 hours
4	Software and firmware and their updating	1. Identify different software platforms and their use 2. firmware loading and adaptation principle	1. Select the developing platform suitable for you 2. load and update the corresponding firmware.	2 hours

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
5	Building of software development environment	1. Build Arduino platform 2. load and update of Arduino library file 3. understand serial communication	1. Familiar with Arduino platform 2. load a library 3. operate and run the first line of codes	2 hours
6	Learning and development of robot library	1. Basic communication and operation types of robots 2. common operating methods of robots 3. control of direction and coordinate modes	1. Communicate with the robot 2. control the robot to move 3. operate the IO interface, gripper, etc. of the robot;	5 hours
7	myBlockly operation robot arm	1. Understand the basic architecture and relation of graphical programming language interfaces: sensor, actuator and procedure 2. variable, cycle and judgment 3. control method of robot arm	1. Display different fonts in the basic 2. make the robot arm move to different positions using three buttons of the basic 3. control the robot arm to make it move to several positions circularly	10 hours

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
8	The use of roboFlow	1. Learn the industrial operating systems commonly used for robots 2. learn the common modules for roboFlow: point, quick movement, IO control and output 3. learn the advanced modules of roboFlow: cycle, judgment, and pallet program	1. Control the movement of the robot arm 2. basic control of IO input and output 3. cycle control and judgment	5 hours
9	Algorithms related to image recognition	1. Common color recognition methods and strategies 2. common shape recognition methods and strategies 3. common area recognition methods and strategies	1. Building of a ROS environment 2. reading of different colors 3. recognition of different shapes	20 hours
10	Vision and the joint debugging of the robot	1. Connect the world with a camera coordinate system 2. QR code image calibration 3. movement and correction	1. Operate the robot arm to the camera coordinate system 2. the robot arm moves in the camera coordinate system 3. recalibrate and set	10 hours

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
11	Artificial intelligence (AI) package	1. Flow chart learning and making 2. electrical connection diagram learning and making 3. operation strategies such as image recognition and classification, etc	1. Sensor connection 2. gripper actuator connection and driving 3. robot arm driving and visual joint debugging	20 hours

4 Additional problems

If the above learning contents cannot meet your actual use needs, you can contact **Elephant Robotics helper** for further communication. We provide customized services for software and hardware, and the service fee is based on the **actual cost**.

mechArm 270-Pi



1 Product Introduction

mechArm 270-Pi belongs to Elephant Robotics's " mechArm " series of small six-axis robotic arms. It adopts Raspberry Pi microprocessor and supports ROS simulation software. It is an industrial imitation structure launched by Elephant Robotics for maker innovation and robot industry-university-research services Small six-axis robotic arm.

The body weight of mechArm 270-Pi is 1kg, the load is 250g, and the working radius is 270mm. The design is compact and portable, small but powerful, easy to operate, and can work with people safely. As the first small six-axis robotic arm of Elephant Robotics, it has three advantages of ease of use, safety and economy, and is a cost-effective choice.

2 Product performance

- **Built-in Raspberry Pi ecology, unlimited development possibilities**
 - Raspberry Pi 4B, 1.5GHz quad-core microprocessor, running Debian/Ubuntu platform.
 - Support 4-way USB, 2-way HDMI, standardized GPIO interface, TF card pluggable.
- **Comes with ROS, graphical programming Blockly**
 - Built-in ROS simulation manipulator operation state, super scalability.
 - blockly visual programming, while supporting general Python software interface.
- **Image recognition rich accessories are widely used**
 - Built-in image recognition algorithm, optional camera can be selected.

- Independently match different accessories such as monitors, gripper suction pumps, etc., to achieve more application scenarios.
-
- **Unique industrial design, extremely compact**
 - Integrated design, the overall body structure is compact, and the net weight is only 960g, which is very easy to carry.
 - Modular design, less spare parts, low maintenance cost, quick disassembly and replacement, and plug and play.
 - **High configuration, with Lego interface**
 - Contains 6 high-performance servo motors, with fast response, small inertia and smooth rotation.
 - The base and the end are equipped with LEGO components, which are suitable for the development of various miniature embedded devices.

mechArm 270-Pi

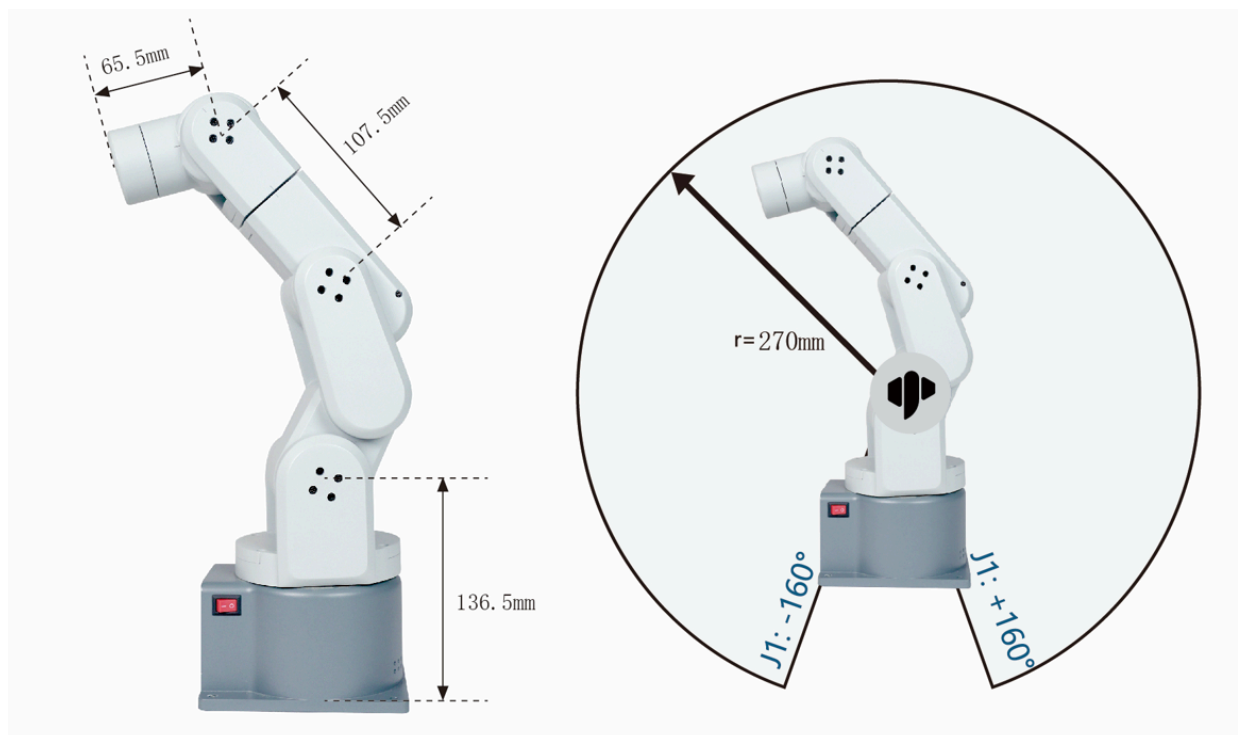


1 Structural parameters

1.1 Robot arm parameters

Indicator	Parameter
Name	Elephant small six-axis robotic arm
Model	mechArm 270-Pi
Degree of freedom	6
Repeated Positioning Accuracy	± 0.5 mm
Load	250g
Dead weight	1kg
Effective working radius	270mm
Power input	8-12V 5A
Motor type	High-precision magnetic encoder motor * six
Movement maximum speed	120°/s
Controller	Raspberry Pi4B

1.2 Scope of work



1.3 Range of motion of joints

joint	scope
J1	-165 ~ +165
J2	-90 ~ +90
J3	-180 ~ +65
J4	-165 ~ +165
J5	-115 ~ +115
J6	-175 ~ +175

1.4 Hole installation

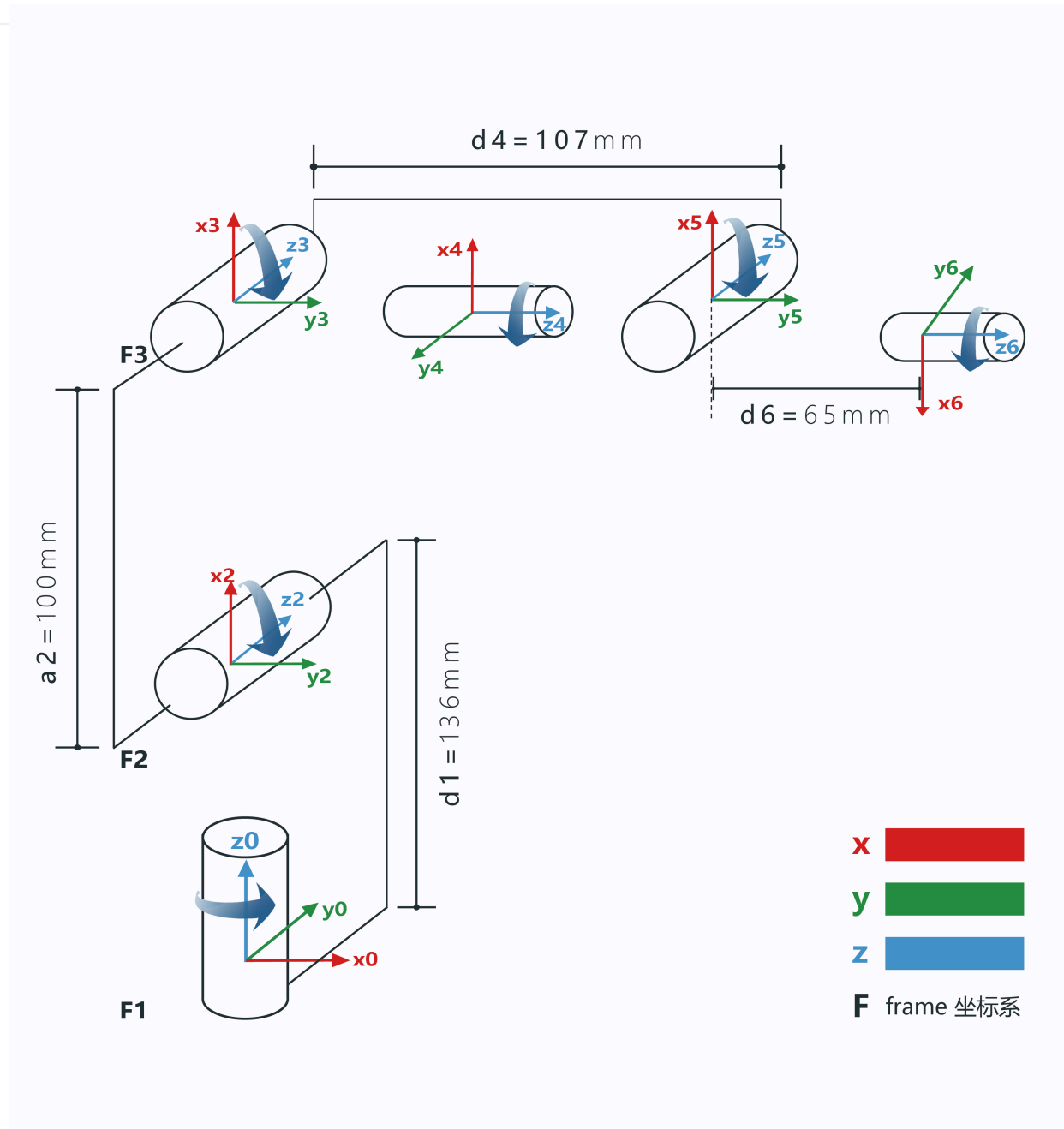
- The Robot Base Mounting Flange Base supports the installation of LEGO components.



- Robot end mounting flange The end of the robot arm supports the installation of LEGO components.



1.5 DH parameter



2 Hardware Parameter

Items	Parameter
Core	240MHz dual core. 600 DMIPS, 520KB SRAM. Wi-Fi, dual mode Bluetooth
Flash	4MB
IO	G19, G21, G22, G23, G25, G33
Bluetooth	2.4G/5G
Wireless type	2.4G 3D Antenna
Core type	M5Stack-basic/Atom

Robotic Arm Electrical Interface

1 Base electrical interface

1.1 Introduction to the base

A, Figure 1-1 shows the front ports and buttons on the base :



Figure 2-3 Pedestal diagram

B. The interface on the left side of the base is shown in Figure 2-4:



Figure 2-4 Left side view of the base

①: Robotic arm charging port: use a DC power socket with an outer diameter of 6.5mm and an inner diameter of 2.0mm; the 8.4V 5A DC power adapter provided by the manufacturer can be used to power the robot arm.

②: Bottom pinout: see table in next paragraph

③: USB2.0 and USB3.0 interfaces: interfaces for data connection with serial bus standards 2.0 and 3.0; users can use the USB interface to copy program files, or use the USB interface to connect peripherals such as mouse and keyboard.

④: Network port: the port for network data connection, the user can use the Ethernet interface for the communication and interaction between the PC and the robot system, and also for Ethernet communication with other devices.

⑤: Switch button: red is switch, I is power on, O is power off.

⑥: Raspberry Pi 4B comes with a USB port

⑦: Display interface: HDMI interface, which is HDMI D-type interface , connected to the display. HDMI interface 2 has priority, and HDMI interface 1 is recommended.

⑧: Speaker, headphone jack

1.2 Description of the bottom electrical interface

Note: Function interface group is 2.54mm Dupont interface, external can use 2.54mm Dupont wire.

A. Figure 1-4 shows the definition of each interface in a functional interface group.

Tag name	Signal name	Function	Note
5V	5V	Power supply, DC5V	
5V	5V	Power supply, DC5V	
GND	GND	Motherboard power signal ground	
NC	NC	--	Temporarily not open
NC	NC	--	Temporarily not open
18	GPIO18	3.3 V-out-PNP output /3.3 V-int input	
GND	GND	Motherboard power signal ground	
23	GPIO23	3.3 V-out-PNP output /3.3 V-int input	
24	GPIO24	3.3 V-out-PNP output /3.3 V-int input	
GND	GND	Motherboard power signal ground	
25	GPIO25	3.3 V-out-PNP output /3.3 V-int input	
08	GPIO08	3.3 V-out-PNP output /3.3 V-int input	
07	GPIO07	3.3 V-out-PNP output /3.3 V-int input	
01	GPIO01	3.3 V-out-PNP output /3.3 V-int input	
GND	GND	Motherboard power signal ground	
12	GPIO12	3.3 V-out-PNP output /3.3 V-int input	
GND	GND	Motherboard power signal ground	
16	GPIO16	3.3 V-out-PNP output /3.3 V-int input	
20	GPIO20	3.3 V-out-PNP output /3.3 V-int input	
21	GPIO21	3.3 V-out-PNP output /3.3 V-int input	
3.3	3.3V	DC3.3 V power supply	
NC	NC	--	Temporarily not open
03	GPIO03	3.3 V-out-PNP output /3.3 V-int input	
04	GPIO04	3.3 V-out-PNP output /3.3 V-int input	
GND	GND	Motherboard power signal ground	
17	GPIO17	3.3 V-out-PNP output /3.3 V-int input	
27	GPIO27	3.3 V-out-PNP output /3.3 V-int input	
22	GPIO22	3.3 V-out-PNP output /3.3 V-int input	
3.3	3.3V	DC3.3 V power supply	

Tag name	Signal name	Function	Note
10	GPIO10	3.3 V-out-PNP output /3.3 V-int input	
09	GPIO09	3.3 V-out-PNP output /3.3 V-int input	
11	GPIO11	3.3 V-out-PNP output /3.3 V-int input	
GND	GND	Motherboard power signal ground	
00	GPIO00	3.3 V-out-PNP output /3.3 V-int input	
05	GPIO05	3.3 V-out-PNP output /3.3 V-int input	
06	GPIO06	3.3 V-out-PNP output /3.3 V-int input	
13	GPIO13	3.3 V-out-PNP output /3.3 V-int input	
19	GPIO19	3.3 V-out-PNP output /3.3 V-int input	
26	GPIO26	3.3 V-out-PNP output /3.3 V-int input	
GND	GND	Motherboard power signal ground	

Description: Figure 1-5 shows other functions of the interface. If other functions are used, the I/O function is unavailable.

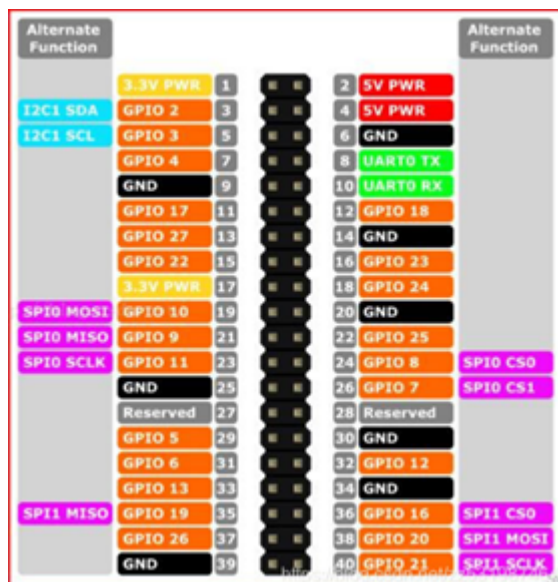


Figure 2-5

B. Power DC interface: The mecharm270 is powered by a 6.5mm od, 2.0mm OD, and a manufacturer's 8.4V 5A DC power adapter.

C. Switch: red is switch, I is power on, O is power off.

D. USB2.0 interface: serial port bus standard 2.0 interface for data connection; The USB port is used to copy program files and connect peripherals such as mouse and keyboard.

E. USB3.0 port (blue) : The port that uses serial bus 3.0 for data connection. The USB port is used to copy program files and connect peripherals such as mouse and keyboard.

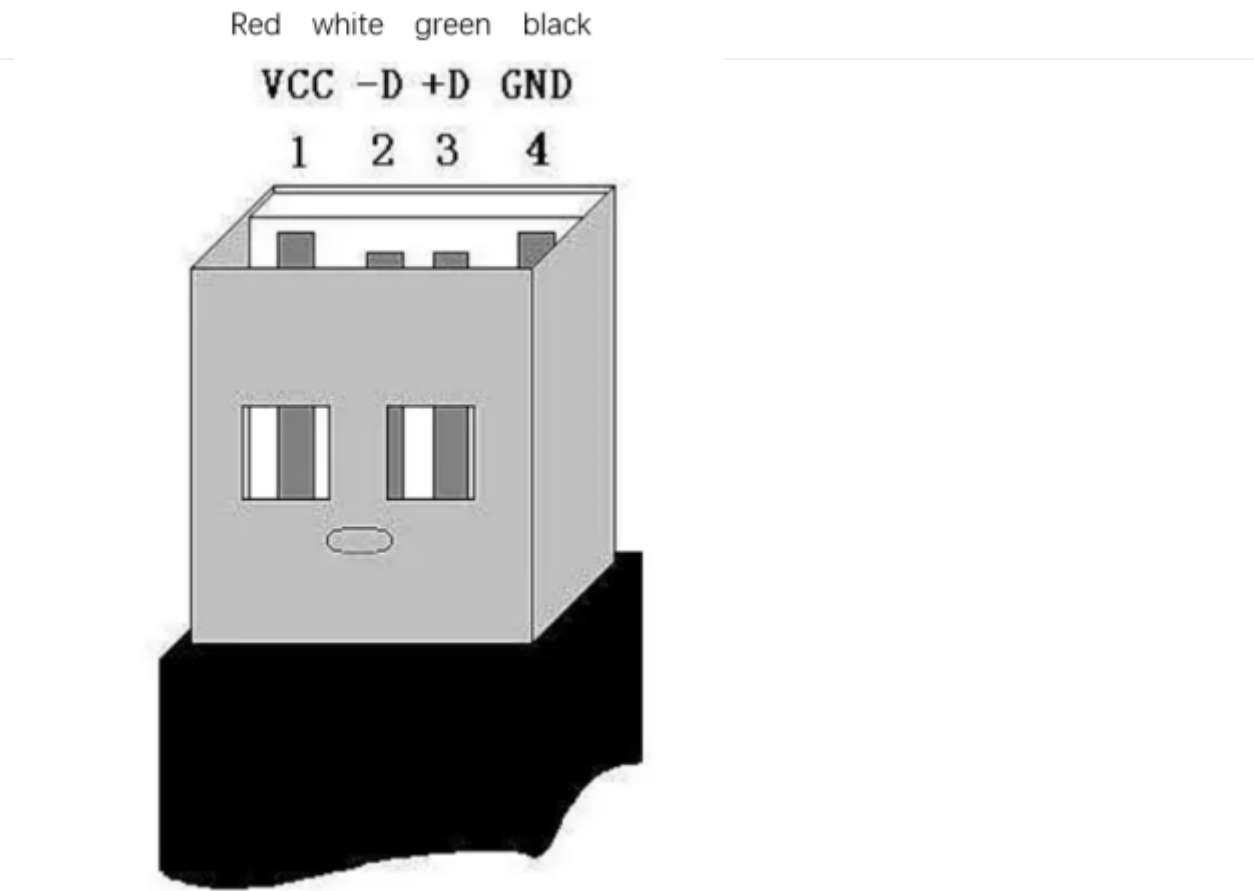


Figure2-6 Defining USB ports

F. The network interface: (as shown in figure 1-7) ports for network data connection. Ethernet interfaces can be used for communication between a PC and a robot system or for Ethernet communication with other devices.

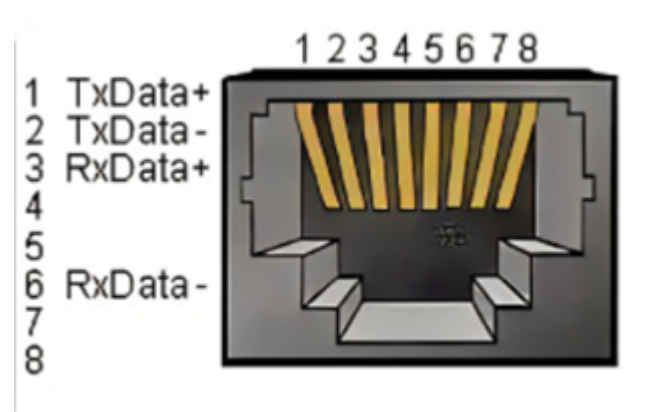


Figure 2-7 Network interface definition

G. HDMI port: The HDMI D-type port (as shown in Figure 1-8,1-9) connects to the monitor. HDMI port 2 has a priority, and HDMI port 1 is recommended.

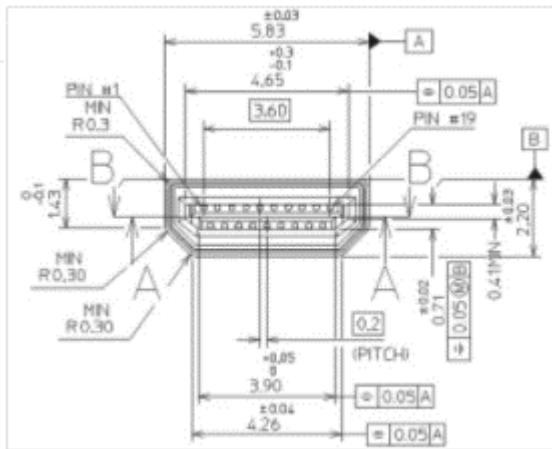


Figure 2-8 HDMI TYPE D port 1

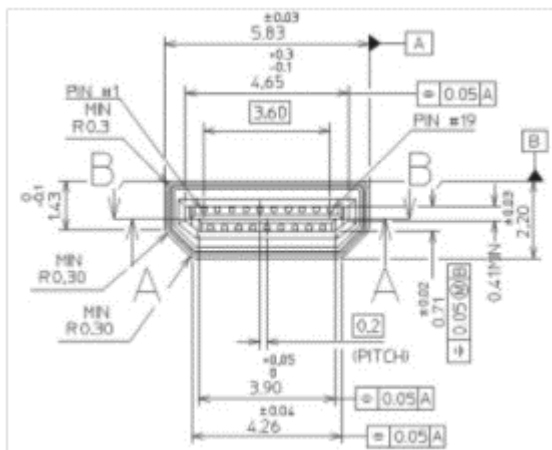


Figure 2-9 HDMI TYPE D port 2

H. Type C port: connects to the PC.

I. SD card socket: an SD card can be inserted and removed. The size of the SD card is 32mm x 24mm x 2.1mm

2 Mechanical arm end electrical interface

2.1 The end of the manipulator is introduced

A. Figure 2-1 show the side interfaces at the end of the manipulator.

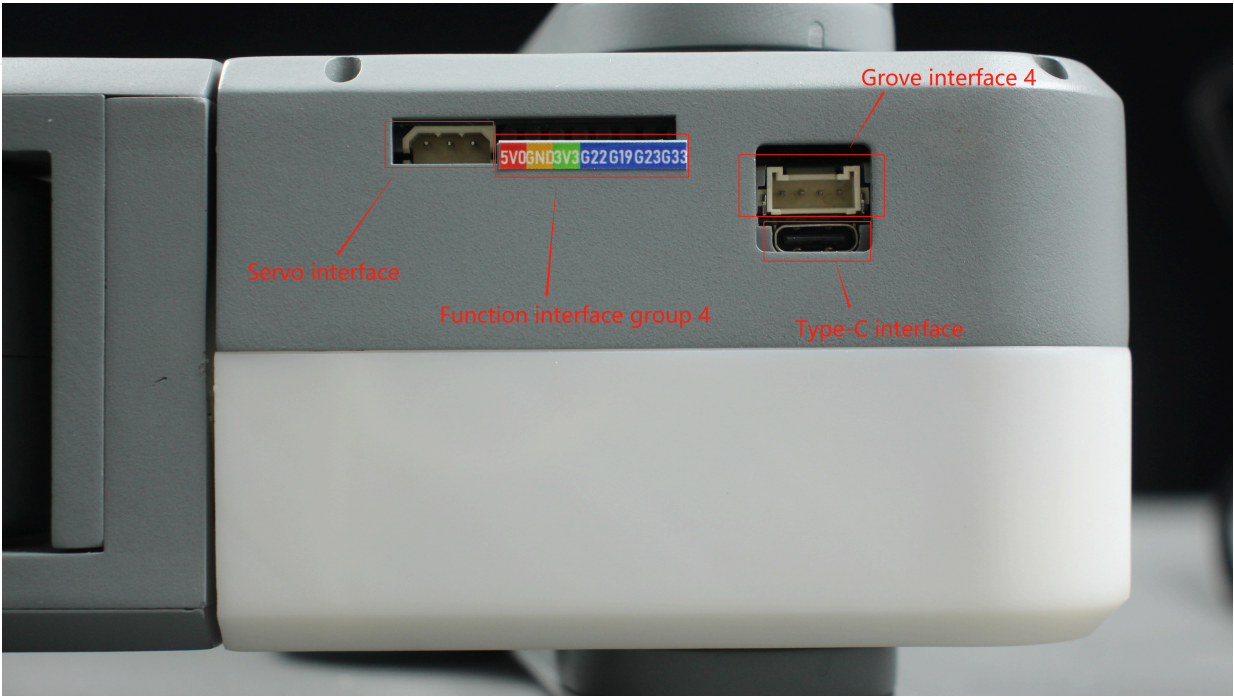


Figure 2-1 End of the robotic arm

2.2 Description of terminal electrical ports

A. Table 2-1 shows the definition of each interface in a function interface group 4.

Tag name	Signal name	Function	Note
5V0	5V	Power supply, DC5V	
GND	GND	Motherboard power signal ground	
3V3	3V3	DC3.3 V power supply	
G22	G22	3.3 V-out-PNP output /3.3 V-int input	
G19	G19	3.3 V-out-PNP output /3.3 V-int input	
G23	G23	3.3 V-out-PNP output /3.3 V-int input	
G33	G33	3.3 V-out-PNP output /3.3 V-int input	

Table 2-1 Functional port group 4

- B. Type C interface: used to communicate with PC and update firmware.
- C. Grove interface 4: Figure 2-2 shows the definition of Grove interface 4

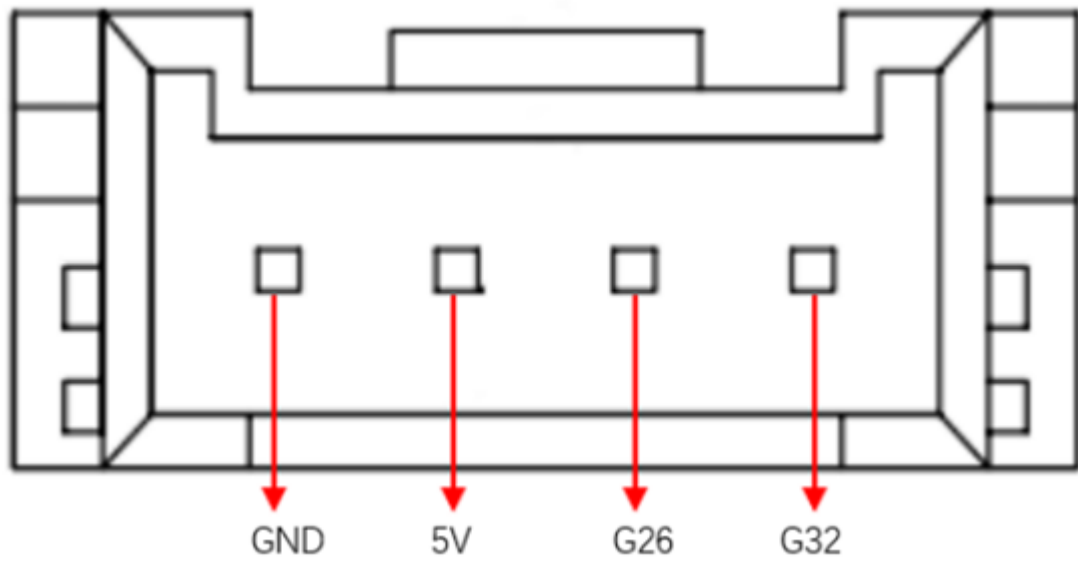


图2-2 Grove port 4

D. Steering gear interface: used for expanding the end of the gripper, currently supporting the use of adaptive gripper.

E. Atom: For 5X5 RGB LED (G27) display and button function (G39)

Introduction

1 Unpacking and working environment

Note : After the packaging box is in place, please confirm that the robot packaging is intact. If there is any damage, please contact the logistics company and the supplier in your area in time. After unpacking, please check the actual items in the box according to the item list.



Table 2-2 mechArm robot arm [standard set]

Contents of the mechArm robot arm [standard set]	-mechArm robotic arm (mechArm 270-Pi) mechArm robot arm-product catalogue -mechArm robot arm-supporting power supply -USB-Type C -jumper --M4*35, cup head socket head, full thread -stainless steel screw-hexagon wrench
--	---

Install the robot system in an environment that meets the conditions described in the table in order to develop and maintain the performance of this machine and use it safely.

Table 2-3 Working Environment and Conditions

working environment	condition
temperature	-10°C~45°C
Relative humidity	20%~70%
Indoor and outdoor requirements	indoor
Other environmental requirements	- Avoid sunlight exposure. - Keep away from dust, oil fume, salt, iron filings, etc. - Keep away from flammable and corrosive liquids and gases. - Must not come into contact with water. - Does not transmit shock and vibration etc. - Keep away from strong electromagnetic interference sources.

2 Installation conditions and requirements

The actual weight of the mecharm robot is 1kg. Considering that during use, as the robot moves, its center of gravity will move, so the robot needs to be fixed on a solid base before it can be used normally. Base weight requirements: fixed base, or mobile base.

Robot base interface size

The base fixing hole is the interface between the fixed robot and other bases or planes. The specific hole size is shown in the figure below, and it is installed with LEGO components.



Please make sure there are corresponding holes on the fixed base before installing.

Before the official installation, please confirm:

- The environment to be installed meets the requirements of Section 2.6.1.3 above.
- The installation position is not smaller than the working range of the robot, and there is enough space for installation, use, maintenance and repair.
- Put the stand in a suitable position.

- Installation related tools are ready.

After confirming the above, please move the robot to the base mounting table, adjust the robot position, and align the fixing holes of the robot base with the holes on the base mounting table. After aligning the hole, press it.

Note: When adjusting the position of the robot on the base installation table, please try to avoid pushing and pulling the robot directly on the base installation table to avoid scratches. When manually moving the robot, try to avoid applying external force to the fragile parts of the robot body, so as to avoid unnecessary damage to the robot.

3 Power on the robot

Before operation, please confirm that you have read and ensured that you have followed the contents of Chapter 1 Safety Instructions to ensure safe operation. At the same time, connect the power adapter to the robotic arm, and fix the base of the robotic arm on the table. The connection method is shown in Figure 3-1.



Figure 3-1 Location of the power connector

mechArm must be powered with an external power supply to provide sufficient power:

- Rated voltage: 8-12V
- Rated current: 5A
- Plug Type: DC 5.5mm x 2.5

Note that you can't just use the TypeC plugged into the M5Stack-basic for powering. Use an official power supply to avoid damage to the robotic arm.

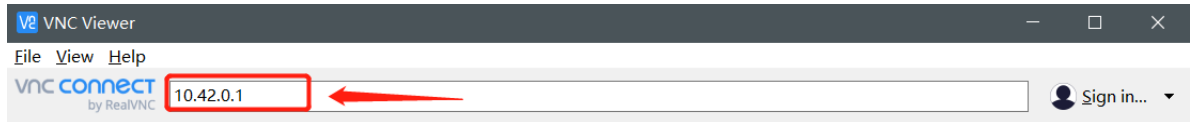
4 Connection Type

Ubuntu (V-20.04)

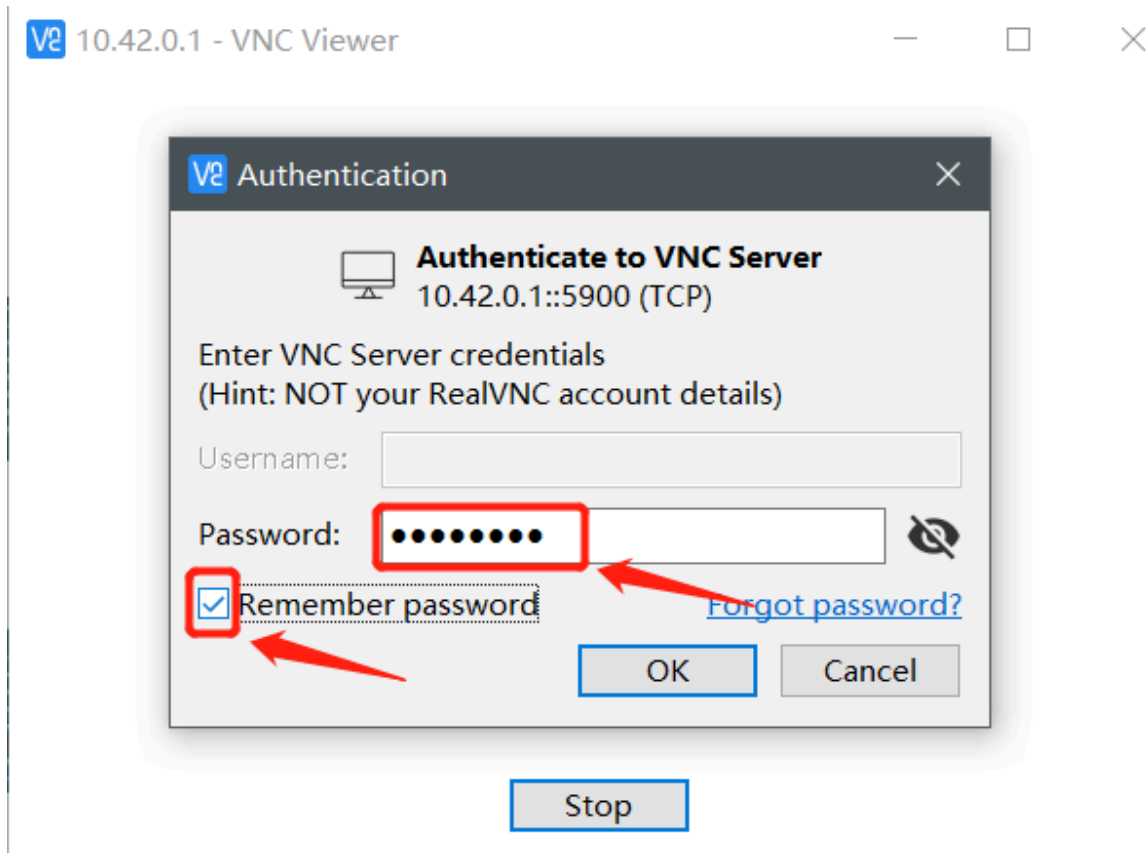
This system has a built-in network which can be directly connected, users can remote control robotics with VNC Viewer. No device such as a monitor is required

- Download [VNC Viewer](#)

- Connect to a wireless network `ElephantRobotics_AP` , password is `Elephant`
- Open VNC and input the IP address of the robotics `10.42.0.1`



- Input VNC password `Elephant`



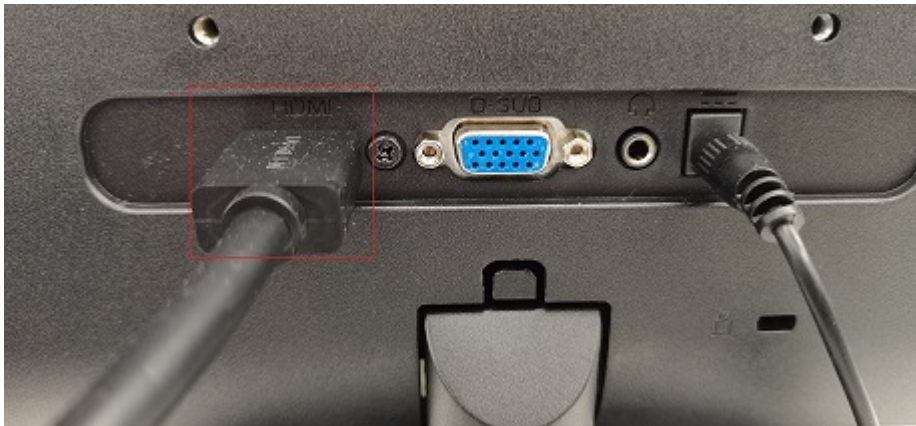
- When you see this page, it means connect successfully



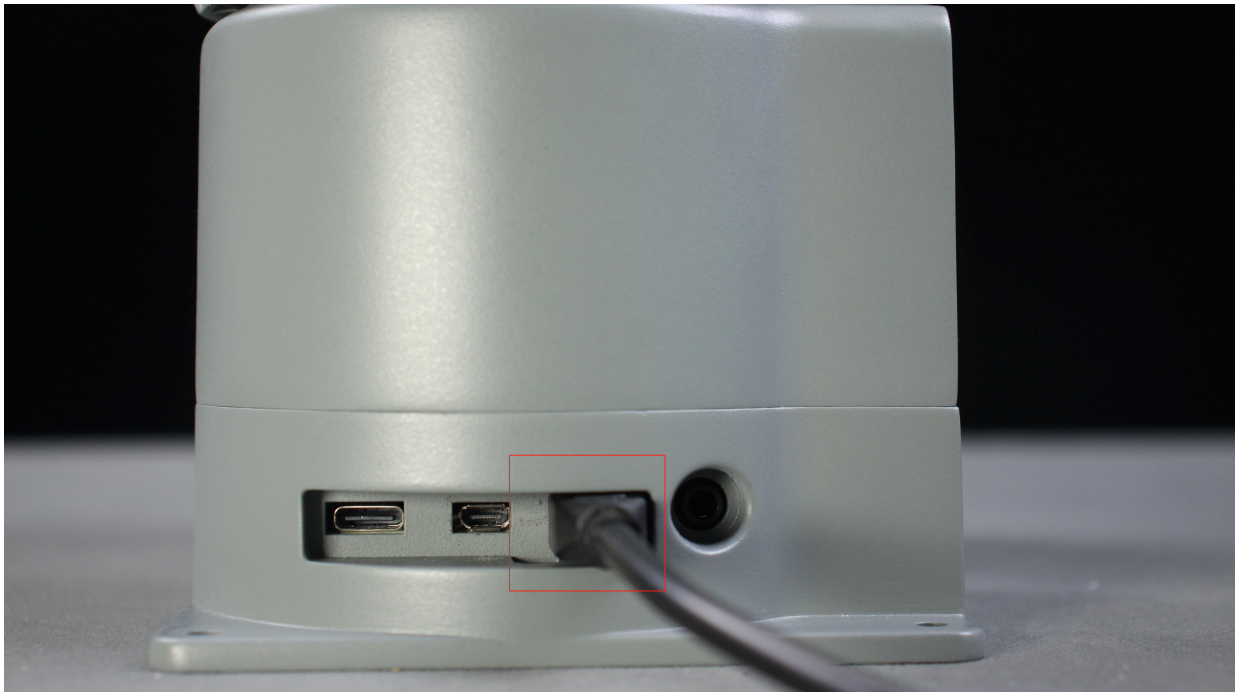
- If you want to control robotics directly, please refer to **Ubuntu (V-18.04)**

Ubuntu (V-18.04)

- Use the matching HDMI cable to connect the robotic arm and the monitor:
- First plug the HDMI cable into the HDMI port of the monitor.



- Then plug the other end into the HDMI port of the robotic arm.



5 Use of end tools

5.1 Suction pump

Details: [4 Adsorption pump](#)

Please scan the code to watch the installation tutorial of the suction pump:



5.2 Adaptive gripper

Details: [5 Adaptive Gripper](#)

Adaptive gripper installation video to be added...

Unboxing vedio



Development environment and construction

1 Use environment

Ubuntu (V-20.04)

mechArm 270-Pi version comes with Ubuntu (V-20.04) system, built-in Python, myStudio, myBlockly, ROS, OpenCV and driver libraries, so there is no need to download development software and install development environment (additional Python libraries need to be installed by yourself), In addition, this system can be directly connected to the built-in network of the robotics and controlled remotely through VNC , without connecting the monitor **.

Ubuntu (V-18.04)

mechArm 270-Pi version comes with Ubuntu (V-18.04) system, built-in Python, myStudio, myBlockly, ROS, OpenCV and driver libraries, so there is no need to download development software and install development environment (additional Python libraries need to be installed by yourself), You only need to prepare a monitor to connect the robotic arm. This system can be [download](#), for details, see Section [15.2 Burning Mirroring](#).

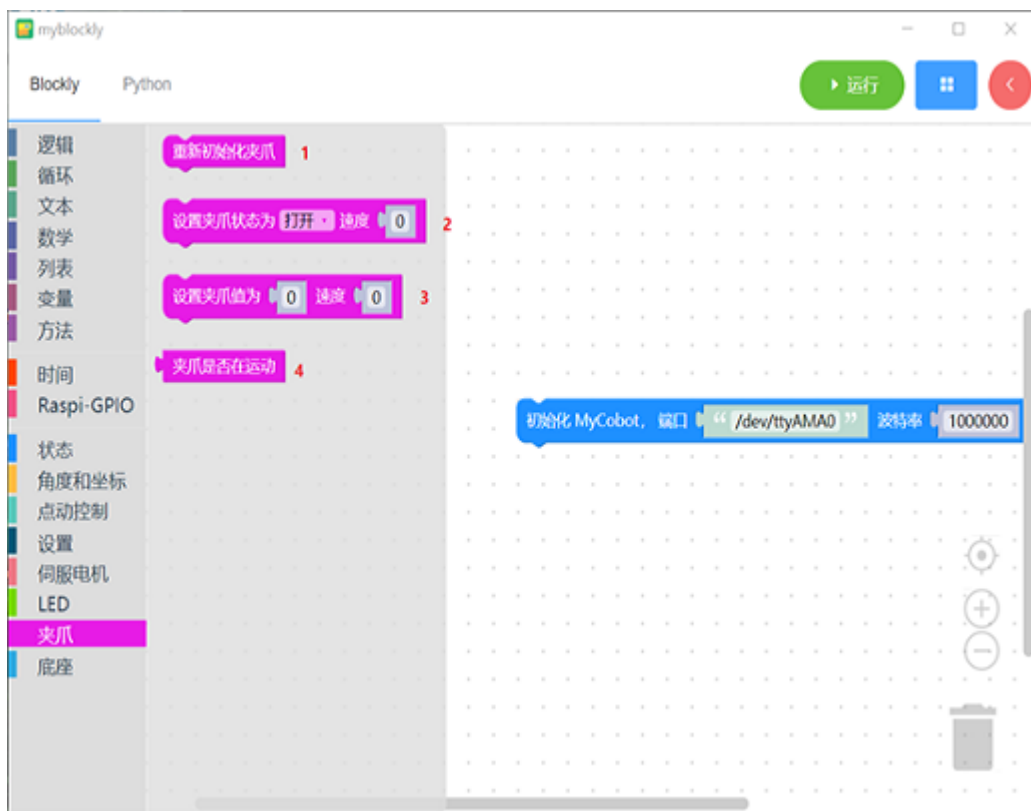
2 Development Environment

Because mechArm 270-Pi has a built-in development environment, it can be used directly. Currently we offer Ubuntu (V-20.04) and Ubuntu (V-18.04).

The following are the development environments supported by the mechArm 270 Pi version and the installation and usage tutorials:

Ubuntu (V-20.04)

- Before using this system, if no device such as the monitor is available, please download [VNC Viewer](#) first. If you have monitor, you don't need to download this software for remote control. You can directly connect the robotic and control it.
- [2.1 Development and use based on Blockly](#) myBlockly embodies the idea of drag-and-drop programming and is a graphical programming software and visualization tool. Users can build code logic by dragging and dropping modules. The process is very similar to building blocks. Can be viewed directly [myblockly use case](#) 。

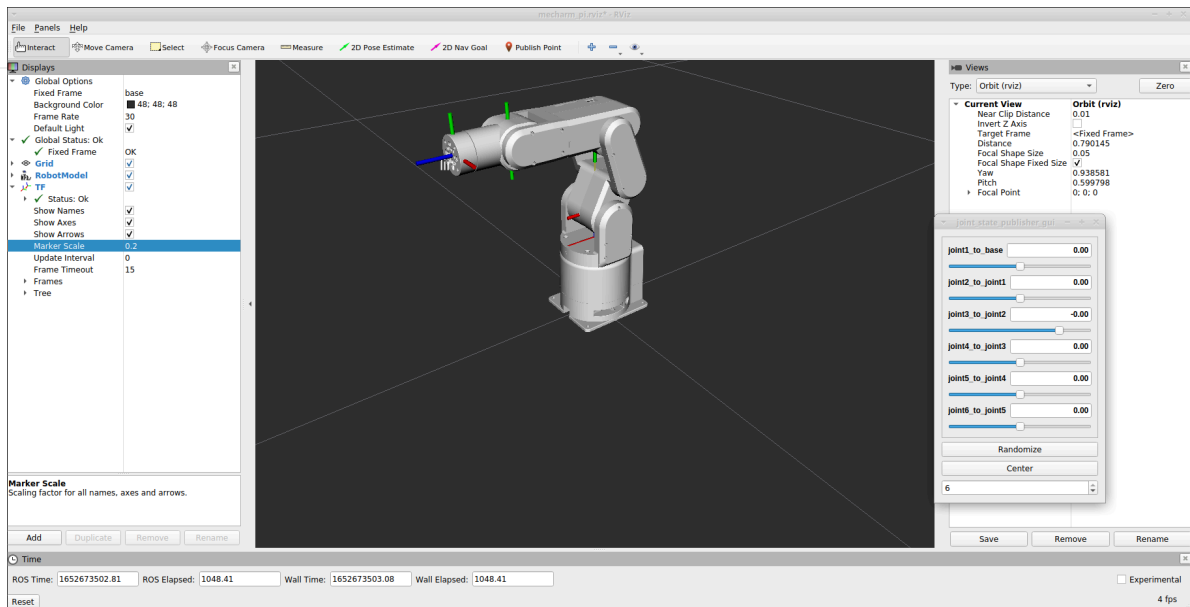


- [2.2 Developed and used based on Python](#) Our products are more friendly to python, and the development of python API library is also increasingly perfect. Through python, the joint angle, coordinates, gripper and other aspects of the robot can be controlled, and there are many choices. Can be viewed directly [use case](#).



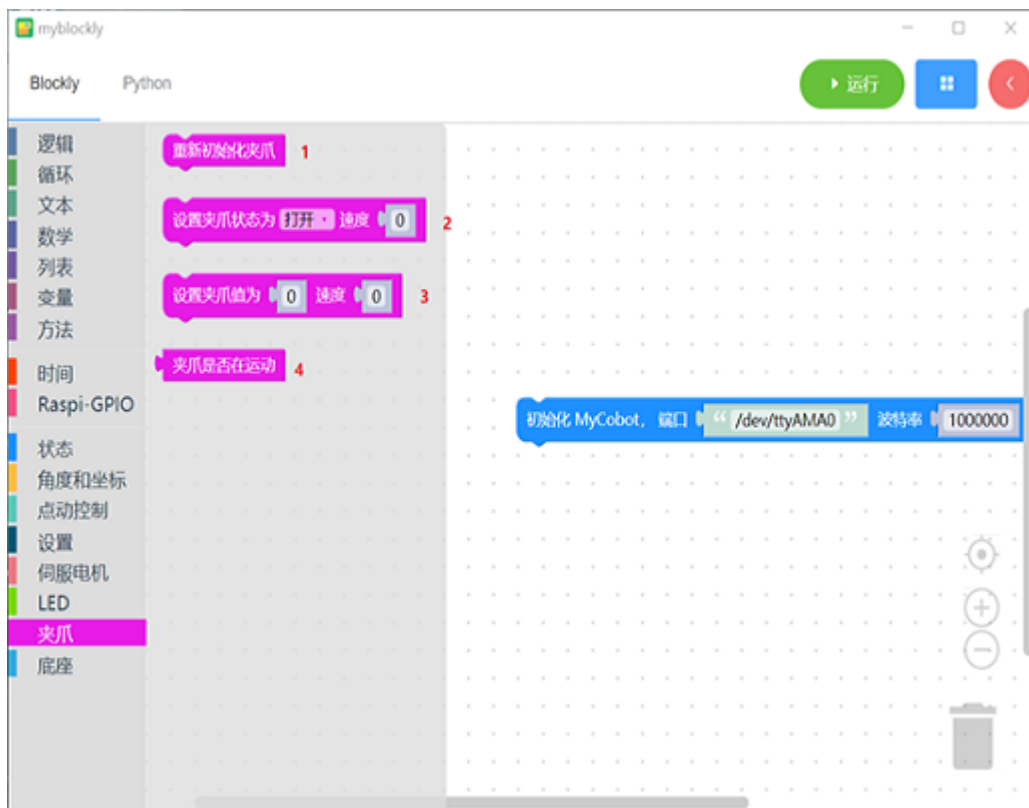
- [2.3 Developed and used based on ROS](#) ROS is open source and a post-operating system, or secondary operating system, for robot control. Through ROS, we can realize the simulation control of the manipulator in the virtual environment. We will visualize the robotic arm through the rviz platform, and operate our robotic arm in a variety of ways; plan and execute the robotic arm's action path through the moveit platform to achieve the effect of freely controlling the robotic arm. Can be viewed directly [Install ROS](#) 后，即可查看 [rviz use cases](#) 、[ROS1 use cases](#) 、[ROS2 use cases](#) 、[use of moveit](#)。



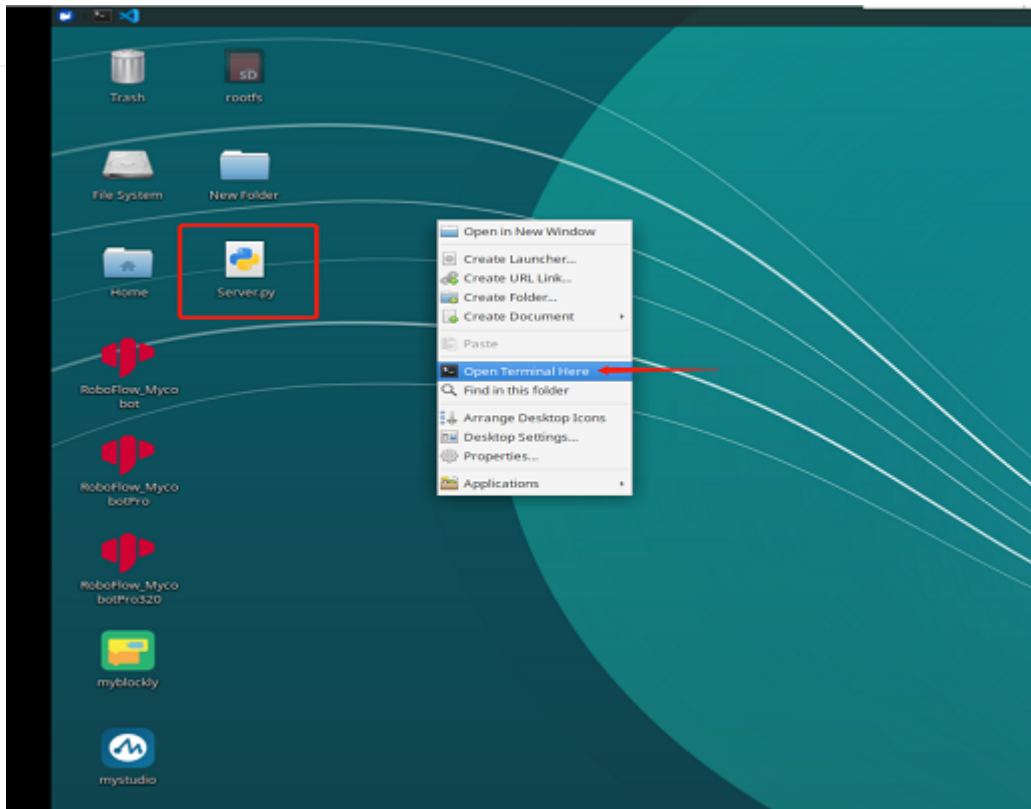


Ubuntu (V-18.04)

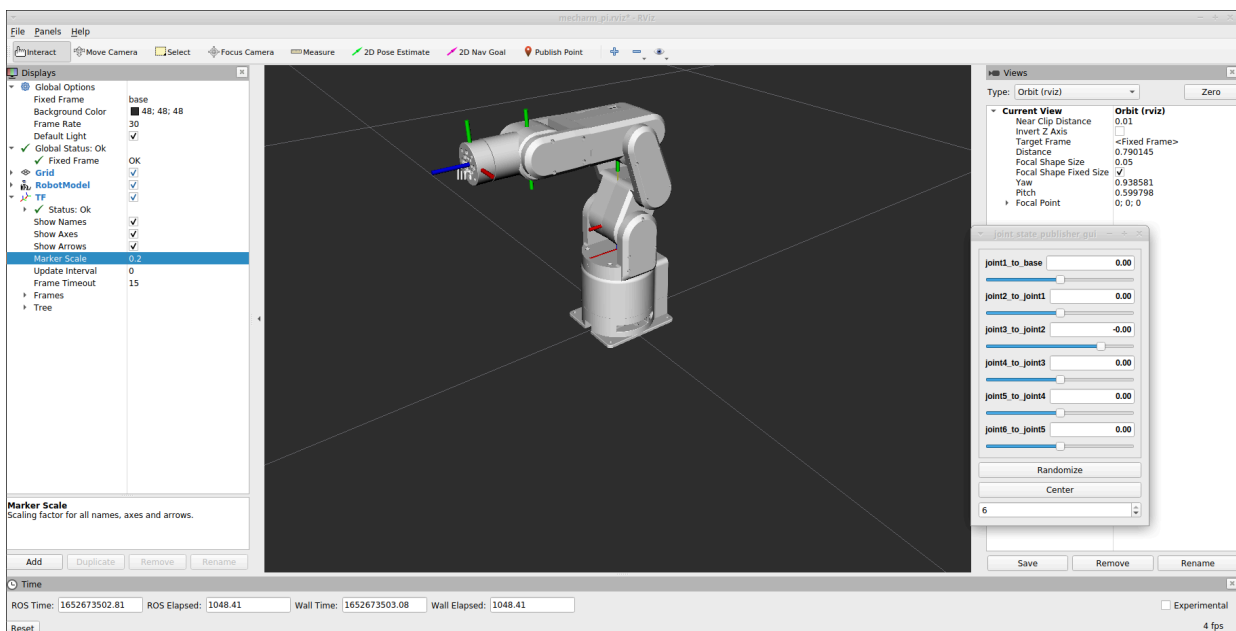
- [2.1 Development and use based on Blockly](#) myBlockly embodies the idea of drag-and-drop programming and is a graphical programming software and visualization tool. Users can build code logic by dragging and dropping modules. The process is very similar to building blocks. Can be viewed directly [myblockly use case](#) .



- [2.2 Developed and used based on Python](#) Our products are more friendly to python, and the development of python API library is also increasingly perfect. Through python, the joint angle, coordinates, gripper and other aspects of the robot can be controlled, and there are many choices. Can be viewed directly [use case](#) .



- **2.3 Developed and used based on ROS** ROS is open source and a post-operating system, or secondary operating system, for robot control. Through ROS, we can realize the simulation control of the manipulator in the virtual environment. We will visualize the robotic arm through the rviz platform, and operate our robotic arm in a variety of ways; plan and execute the robotic arm's action path through the moveit platform to achieve the effect of freely controlling the robotic arm. Can be viewed directly [use case](#) 和 [use of moveit](#).



3 Firmware update

The Atom firmware of mechArm 270-Pi need to be updated using myStudio. mechArm 270-Pi built-in myStudio, which can be used directly.



Installation and usage tutorial of myStudio:

- 4.1 myStudio
 - 1 Install the driver
 - 2 Updating device firmware

Common Problem

In this part, some common driver-related problems, software-related problems and hardware-related problems are listed.

How To Ask Questions Gracefully [1 How To Ask Questions Gracefully](#)

Drive Related [2 Drive Related](#)

Software Problem [3 Software Problem](#)

Hardware Problem [4 Hardware Problem](#)

If you have purchase intention or any parameter questions, please send an email to this mailbox.
sales@elephantrobotics.com

If the listed problems can't help you solve and you have more after-sales questions, please send an email to this mailbox. support@elephantrobotics.com

How To Ask Questions Gracefully

1 When asking questions in various places, you will find several phenomena:

- No answer after asking question.
- It took a long time for the question to be answered.
- The other party always despise me and do nothing.

2 Before asking questions, make sure you have studied chapters three or four.

Chapters **3 and 4** of this document are the basis for using the **my series**. No matter whether you have development experience or not, be sure to read and operate from front to back.

Many problems will be solved during this process. Do not ask questions in QQ groups, forums, issues, or emails at the very beginning. Many problems explained in the document at the beginning may not be answered in a timely by the community. To save everyone's time, and for a better community environment, everyone can grow together better, please understand each other.

3 When asking questions, try to do the following, which will greatly increase the chances of a quick resolution:

To figure out what's going on and what I did, including:

- What effect and what function do I want to achieve?
- In order to achieve this effect, how do I do it and what is the detailed process?
- In the process of implementation, what error occurred and what is the phenomenon (for example, what is the error reported, what is the **complete** error content?)
- Have I read the error message carefully, and is there any indication of the cause and solution of the error in the error message?
- Based on these error messages, think carefully, can I solve the problem?
- Can I find a solution to the problem by searching the documentation, issues, and using a search engine?

4 If you really can't solve the problem yourself, you need to ask someone for help, and you need to consider:

- Who to ask, where to ask, and who has a better chance of answering my question? And how about real-time?
- What data and phenomena should I give him so that he is willing to help me solve the problem quickly?
 - Provide my purpose (to let the answerer know what you are doing).
 - Provide the complete implementation process and the phenomena that occur in the process (for the answerer to follow your process to do it again, that is, the problem recurs).
 - Give the wrong place, indicate where the phenomenon or result is different from what you expected! (Let the answerer know, where did not meet expectations).
 - For the error information that appears, it needs to be complete, as many screenshots as possible, more logs, don't be stingy to take a small picture, or give a part of the log (because the answerer may not have

done this for a long time, they forgot some details, and they need to rely on screenshots and complete logs to quickly recall. And according to the detailed logs, they can quickly locate where the problem is).

- How to ask questions with a more sincere attitude, even if I don't understand anything, everyone is willing to answer.

5 Question Template

Try to ask questions as elegantly as possible, without adding redundant modal particles, complaining vocabulary, considering every word and punctuation, thinking about the problem from the perspective of the answerer, and how to let the answerer help me solve the problem quickly. Too few words will make it difficult to describe the question, and too many words will make the answerer impatient.

6 Title

No matter where the question is asked (including QQ group), prepare a title of about 30 words for your question, clarifying the central idea of the question, including:

- It is necessary to clearly distinguish the type of problem, whether it is a question to ask, a bug submission, or an experience sharing and so on. Let answerer instantly locate what you want to do on a screen full of text.
- One sentence to clarify the core of the problem, such as run the camera sample program, report an error reset fail, it may be a hardware problem.

So the title after synthesis can be like this:

- [Mycobot question] Running the camera sample program, the error reset fail is reported. Could it be a hardware problem?

Try to **don't** appear in such a title:

- Why is my board not working again?
- Why is my code not working?
- Why is my screen black?
- [Mycobot question] I received the development board, why is the development board screen red and has a small line of text?
- I ran a program and something went wrong.

You can ask this:

- [Mycobot question] My board can't start after I connected the power supply in reverse, how can I tell where the board is burnt, and if so, how can I save it?

7 Content

First stand on the answerer's point of view, if asked a question:

- First of all, I need to know what the other party wants to do and what the goal is to achieve.
- In order to achieve this goal, what steps did he refer to?
- In fact, what specific steps were used, and then at which step the problem occurred, so that I can try to reproduce the phenomenon according to his steps. If this problem seems to be difficult to solve and there are no steps to reproduce it, it may take a lot of time to reproduce, so let's put it aside and solve other problems first.
- What was the specific problem that arose, and if he only stated the problem, how would I know what was wrong with him, maybe a physical discomfort? So this is very important, I need to ask him to explain the phenomenon of the problem and indicate what is different from the expected, otherwise I have to guess what is the difference between the comparison and the expected, and the time to solve the problem has increased.

- If there is a problem, I may need his log file, so that I can analyze the source code according to the log, otherwise it may be difficult to solve the problem, then this problem can be looked at later.
-

In summary, you can ask the following questions:

- Elaborate on your goals, what you want to do, and what the phenomenon should look like.
- Is there any documentation, code, or tutorial I refer to?
- How to reproduce the error: how to do it in detail, write each step in detail until the problem occurs.
- Elaborate on what happened when the error occurred, and how it was different from what was expected, and needed to prove that the problem did occur.
- Attach log files, as well as screenshots, or even videos, logs and screenshots must be complete, not just a small part, the answerer may find some problems you did not notice from your full log and screenshots, this is very important.
- In addition, pay attention to the format when pasting the code. Do not display it in a mess after pasting, and it cannot be seen. Try to copy it and run it directly.
- Finally, you need to thank the community friends who answered the question.

Drive Related

1 About python

Q: `send_coords([x,y,z,rx,ry,rz], speed, 1)` What do the parameters in this API mean? What do rx, ry, and rz correspond to Euler angles? What is the rotation order of Euler angles? And what is the value range of each parameter?

- A: The parameters in the previous array are the coordinates of the end of the myCobot, speed is the speed, and the last parameter is the motion mode. rx, ry, and rz should correspond to rpy, that is, corresponding to roll, pitch, and yaw respectively. The order of Euler angles is zyx, and zyx is its own coordinate. The value range of x, y, z is -300~300.00 (the value range is undefined, if the range is exceeded, the inverse kinematics no solution prompt will be returned), and the value range of rx, ry, and rz is -180~180.

Q: Are sample tutorials for the python API provided?

- A: Yes, there is test code in the test folder of github, which can be executed with the terminal.
<https://github.com/elephantrobotics/pymycobot/tree/main/demo>

Q: How does the python drag teaching demo of mycobot280-Pi work?

- A: Run in the terminal, please enter 1000000 baud rate.

Q: Mycobot280-Pi uses python zero to calibrate the demo program, why is there an error?

- A: The atom firmware is not burned, please burn the atom firmware before running the program.

Q: Is the python API of different versions of myCobots the same?

- A: The API is the same.

2 About ROS

Q: How do a microcontroller-based myCobot and a microprocessor-based myCobot run ROS?

- A: The use of ROS for a microcontroller-based myCobot is currently on Ubuntu, and you can also develop your own ROS. A microprocessor-based myCobot have its own ROS environment and can be used directly.

Q: Can a microprocessor-based myCobot connect to a PC to use ROS and moveit?

- A: The current open source data is not controlled by direct communication. It can be implemented by modifying the existing node files through ros + socket.

Q: Can you provide files and programming examples of the rviz model?

- A: It is available on our github. https://github.com/elephantrobotics/mycobot_ros

Q: Can myAGV create maps remotely?

- A: You can create maps remotely. We are the control interface of open source ROS.

Q: Why is the error permission denied: '/dev/ttyUSB0' reported when using ROS to start the rviz model file?

- A: It is because the serial port permission is not given. You should type `sudo chmod 777 port name` in the terminal.

Q: Why is the error `init()` takes exactly 2 arguments (3 given) reported when running the slider control and model follow commands of ROS?

- A: The pymycobot library is not installed and started.

Q: Q: When using ROS, why is the myCobot angle inconsistent with the model angle after opening the rviz model?

- A: It is very likely that the zero position of the myCobot is not calibrated, and the zero position of the myCobot needs to be calibrated.

Software Problem

Q: Why can't my compiler find the corresponding device?

- A: You need to build a development environment and install the corresponding project library before you can develop the device.

1 About mystudio

Q: What is mystudio?

- A: It is the firmware burner.

Q: Why can't the device run normally after I burn the firmware to the ATOM terminal?

- A: The firmware of the ATOM terminal needs to use our factory firmware. Other unofficial firmware cannot be changed during use. If the device accidentally burns other firmware, you can use the "myCobot firmware burner" to select the ATOM terminal - select the serial port - select the ATOMMAIN firmware for ATOM terminal for burning.

Q: Can the drag teaching in the minirobot firmware control the gripper?

- A: It is achievable.

Q: Why can't drag teaching after the minirobot firmware is burned?

- A: First, check whether the M5Stack-basic firmware and atom firmware have been burned, whether the burned firmware corresponds to the requirements to be realized, and whether the burned firmware is the latest version. The bottom M5Stack-basic burns the minirobot, and the top atom burns the atommain.

Q: What should I do if myCobot's serial port is not recognized on mystudio?

- A: If your computer equipment does not prompt for the connected myCobot, please install the serial port driver first.

Q: Can the track recorded by drag teaching be saved to the card?

- A: Currently it cannot be saved to the memory card. And drag teaching can only save one path at a time, and the next recording will overwrite the previous action.

2 About Roboflow

Q: Can robotstudio software be used for programming?

- A: Our own industrial programming software roboflow can be used. RobotStudio is owned by ABB.

Q: What is the reason for the Quickmove of the roboflow software beyond the limit?

- A: It may be that a joint or multiple joints exceed the limit.

Q: How does roboflow load the already written program?

- A: After logging in, select program robot and click load program. Directly clicking run program cannot be used, only pro600 can.

Q: When the pro600 uses roboflow, the log shows that 456 joints are stopped. Is this normal?

- A: This is normal.

3 About mycobot phone controller

Q: What version of firmware should myCobot phone controller app be programmed with?

- A: You need to burn the atom firmware version atommain 2.5 in mystudio.

4 About myblockly

Q: Why does a pop-up box always appear when myblockly is running?

- A: Before running the myblockly program, close the serial port occupation.

5 About ROS1

Q: When a terminal switches to `~/catkin_ws/src` and uses git to install and update `mycobot_ros`, it appears that the target path "mycobot_ros" already exists. Why?

- A: A `mycobot_ros` package already exists in `~/catkin_ws/src`, so you need to delete it beforehand and run git again.

Q: When rosrn is running, a terminal error is reported saying `could not open port /dev/ttyUSB0: Permission: '/dev/ttyUSB0'`. Why?

- A: The serial port permission is insufficient. Enter `sudo chmod 777 /dev/ttyUSB0` to grant the permission.

Q: Why can't ros programs run in vscode?

- A: Since the vscode terminal cannot be loaded into the ros environment, it needs to run on the system terminal.

Q: Unable to register with master node [http://localhost:11311]: Unable to register with master node. master may not be running yet. Will he keep trying ?

- A: Before running the ros program, start the ros node and enter `roscore` on the terminal.

Q: When rosrn runs, the terminal error shows `could not open port /dev/ttyUSB0: No such file or directory: '/dev/ttyUSB1'`. Why?

- A: The serial port is incorrect. Check the actual serial port of the manipulator. To view the value, run `ls /dev/tty*`.

Q: Failure in Ubuntu18.04 `catkin_make` building code, terminal prompt `Project 'cv_bridge' specifies '/usr/include/opencv' as an include dir, which is not found`. Wait for an error message

- A: The opencv path in the configuration file is inconsistent with the actual path in the system. Need to use sudo to modify the configuration file (path is `/opt/ros/melodic/share/cv_bridge/cmake/cv_bridgeConfig.cmake`), the system actual opencv path in the `/usr/include/` directory.

Q: I cloned the mycobot_ros package and ran the rosrn program directly. package 'mycobot_280' not found or can't find the file?

A: The newly cloned mycobot_ros needs to build code to compile to the ros environment. The input terminal


```
cd ~/catkin_ws/  
catkin_make  
source devel/setup.bash
```

Q: After compiling, why does the following error occur when a new terminal runs launch command?

```
u20@u20-VirtualBox:~/catkin_ws$ roslaunch mybuddy_socket slider_control.launch  
RLException: [slider_control.launch] is neither a launch file in package [mybuddy_socket] nor is [mybuddy_socket] a launch file name  
The traceback for the exception was written to the log file
```

- A1: The system has not added ros environment variable, so every time a new terminal is started, the source is required:

```
cd ~/catkin_ws/  
source devel/setup.bash
```

- A2: ros environment variable is added to the system. There is no need to execute source after starting a new terminal:

```
# The noetic is Ubuntu20.04 system  
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

- A3: The file name in the instruction may not be the same as the file name in the mycobot_ros package. Please carefully check whether the instruction is incorrect.

Questions about the Structure

Q1: What are the maximum angle of Joint 1 and Joint 5 of myCobot?

A1: The maximum angle of Joint 1 and Joint 5 are 160° both clockwise and anticlockwise. Remember to rotate the joints gently. Do not force a rotation when the joint reach its maximum angle.

Q2: What controls the six steering engine?

A2: They are controlled by Atom at the top of the robot.

Q3: What functions Atom performs?

A3: It serves to control the robot by algorithm, such as forward, inverse kinematics and coordinate switching. Atom is temporarily not open-sourced.

Questions about the Communication

Q1: Why the screen has no display although the robotic arm has been connected with HDMI cable? Does it need to download a port driver?

A1: Check whether the connection is correct and the power has been turned on. Try to use another interface and plunge into the interface stably. There is no need to download a port driver.

Q2: What are the versions of communication interface supported by different robotic arms?

A2: The robotic arms of micro-processor supports TCP communication, and the robotic arms of micro-controller supports USB-port communication.

Q3: How much is the frequency of the communication of robotic arm?

A3: 10-20Hz.

Questions about Parameters

Q1: What's the unit of measurement of the speed of robotic arm?

A2: 180°/s.

Solutions to Hardware Problems

1. How to deal with the shaking of robotic arms?

Step 1: Burn the latest version of ATOM via myStudio.

Step 2: Upgrade pymycobot. Click on Win+R, and type cmd to enter in the terminal. Type pip install pymycobot --upgrade --user and then click on Enter.

Step 3: Go to [GitHub](#) to download pid_read_write.py. Reset each parameter of the steering engine according to the prompt message. And then, operate the system again.

Notice: The parameters of each version of robot are different. Set the parameters according to the data given in the pictures below.

```

25  # 参数对应地址 corresponding parameter
26  data_id = [21, 22, 23, 24, 26, 27]
27  # 修改后的参数 reset parameters
28  data     = [32, 8, 0, 0, 3, 3]
29

```

Parameter	Connotation	myPalletizer 260			
		J1/J3	J1/J3	J2	J4
21	location loop P ratio coefficient	25	32	32	32
22	location loop D differentia coefficient	25	32	32	32
23	location loop I integral coefficient	0	0	0	0
24	minimum starting force	0	16	3	0
26	clockwise insensitive area	3	1	1	0
27	anticlockwise insensitive area	3	1	1	0

Parameter	Connotation	mechArm 270			
		J1/J3	J2	J4/J6	J5
21	location loop P ratio coefficient	32	32	25	25
22	location loop D differentia coefficient	8	8	25	25
23	location loop I integral coefficient	0	0	1	1
24	minimum starting force	0	0	0	0
26	clockwise insensitive area	3	1	3	3
27	anticlockwise insensitive area	3	1	3	3

Parameter	Connotation	myCobot 280		
		J1-J3	J4	J5-J6
21	location loop P ratio coefficient	32	25	25
22	location loop D differentia coefficient	8	25	25
23	location loop I integral coefficient	0	1	1
24	minimum starting force	0	0	0
26	clockwise insensitive area	3	3	3
27	anticlockwise insensitive area	3	3	3

Parameter	Connotation	myCobot pro 320		
		J1、J2、J3	J2	J5-J6
21	location loop P ratio coefficient	32	32	25
22	location loop D differentia coefficient	8	8	25
23	location loop I integral coefficient	0	0	1
24	minimum starting force	0	0	0
26	clockwise insensitive area	3	3	3
27	anticlockwise insensitive area	3	3	3

Questions about Gripper & End

Q1: Can the gripper close completely?

A1: Due to the thickness of the gripper, it cannot close completely. You resort to a shim to make it completely closed.

Q2: What is the kind of communication of adaptive gripper?

A2: It is TTL communication.

Q3: What is the communication of the end of myCobot 320?

A3: 485 communication interface.

Q4: How to fix a camera at the end of a robotic arm?

A4: A flange is required to fix a camera.

Other Questions

Q1: Why the electric engine powers off during usage?

A1: Because the engine is too hot to be used. Please wait for a few minutes and reuse again.

Q2: Does the robotic arms support the development based on Android?

A2: We doesn't support Android development. If you want to develop it by yourself, we can provide you with port protocol.

Q3:What is the function of the USB carried by the PI version?

A3: It serves to charge the robotic arm.

Background knowledge

Assessing Prior Knowledge



It is necessary to have an in-depth understanding of myCobot from the aspects of hardware, software and robot algorithm. At the beginning, we can understand the progress of robots by understanding the history of robots.

- [Robot](#)
- [Electric](#)
- [Motors](#)

Robot

This chapter is excerpted from Introduction to robotics mechanics and control by J.Craig. If you want to read more about it, please buy it online.

1 Background

The history of industrial automation is characterized by the rapid renewal of technological means. The renewal of such automation technology is closely related to the world economy, whether as an inducement or a result of the development of the world economy. Industrial robot in the 1960s is undoubtedly a unique equipment, it will be combined with the computer aided design (CAD) system, computer aided manufacturing (CAM) system application, this is the modern manufacturing automation of the latest development trend. These technologies are leading the transition to a new field of industrial automation.

Manipulator is one of the most important types of industrial robots. Whether the manipulator can be called an industrial robot is controversial. The equipment shown here is generally considered to belong to the category of industrial robots, while CNC (NC) grinders are usually outside this category.

Generally speaking, the research of manipulator mechanism and control theory is not a new science, it is only a synthesis of traditional theory. Mechanical engineering theory provides a methodology for the study of manipulator in static and dynamic environments. The mathematical method is used to describe the spatial motion of the manipulator and its characteristics. Control theory provides various design methods and evaluation algorithms for the realization of desired motion or force. Electrical engineering technology can be used in sensor and industrial robot interface design; Computer technology provides the programming platform needed to perform the desired task.

2 Basic Concepts

Mechanical Arm

Mechanical Arm can also be called industrial robot, cooperative robot, manipulator arm, bionic arm, series robot, etc.

Position & Pose

In robot research, we usually study the position of objects in a three-dimensional space. The objects referred to here include not only the lever, parts and grasping tools of the manipulator, but also other objects in the workspace of the manipulator. Usually these objects can be described by two very important properties: position and pose. Naturally, we will first study how to express and calculate these parameters mathematically.

In order to describe the position and posture of a space object, we usually place the object firmly in a space coordinate system, that is, the reference frame, and then we study the position and posture of the space object in this reference coordinate system.

Direct Kinematics

Kinematics is the study of the motion of objects without regard to the forces causing such motion. In kinematics, we study higher-order derivatives of position, velocity, acceleration, and position variables with respect to time or other variables. Thus, the research object of manipulator kinematics is all the geometric and temporal characteristics of motion. Almost all manipulators are composed of rigid links, adjacent links connected by joints that allow for relative motion. If it's a revolute joint, its displacement is called the joint Angle. These joints are usually fitted with position sensors to measure the relative position of adjacent bars. If you have a revolute joint, this displacement is called the joint Angle. Some manipulators have sliding (or moving) joints, so the displacement of two adjacent links is a linear motion, which sometimes called the joint offset.

A typical problem in the study of manipulator kinematics is manipulator forward kinematics. It is a static geometry problem to calculate the position and posture of its end-effector of the manipulator. Specifically, given a set of values for joint angles, the forward kinematics problem is to compute the position and posture of the tool coordinate system relative to the base coordinate system. In general, we refer to this process as the representation of manipulator position from joint space description to Cartesian space description.

Degree of Freedom (DOF)

The number of DOF is the number of manipulator position variables in the coordinate system (reference frame) with figure 1-5 in the manipulator, which determines the position of all components in the mechanism. DOF is universal to all mechanisms. For example, a four-bar mechanism has only one DOF (although it has three movable rods). For a typical industrial robot, the number of joints is equal to the number of DOF because manipulator arms are mostly open motion chains and each joint position is defined by an independent variable.

End-effector

End-effector is installed at the free end of the manipulator. Depending on different applications of robot, it may be a fixture, a welding torch, an electromagnet, or some other devices. We usually describe the position of the manipulator in terms of a tool coordinate system attached to its end-effector, and the corresponding tool coordinate system is the base coordinate system connected to the fixed base of the manipulator.

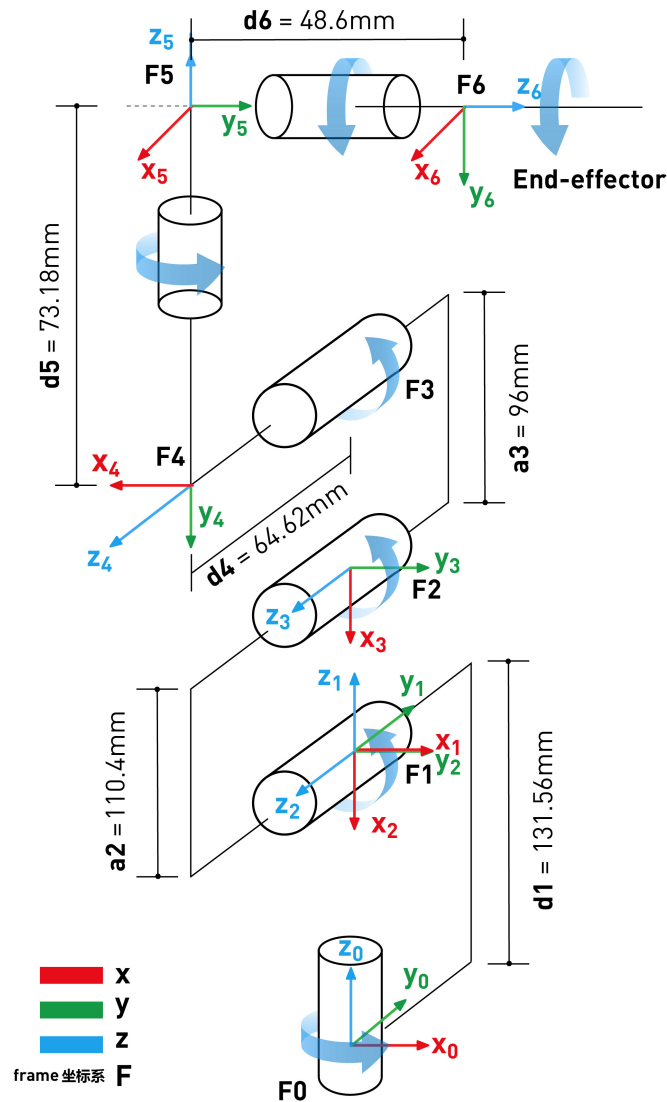
Inverse Kinematic

Given the position and posture of the end-effector of the manipulator, calculating all joint angles that can reach the given position and attitude.

3 Space description

Position

Once the coordinate system is established, we can locate any point in the world coordinate system with a position vector of 3×1 . Since many coordinates are often defined in the world coordinate system, a piece of information must be attached to the position vector indicating which coordinate system is defined. In this book, the position vector has a leading superscript to indicate the coordinate system to which it refers.



Posture

We find that it is often necessary not only to represent points in space, but also to describe the posture of objects in space. For example, if the vector "P" in Figure 2-2 directly determines a point between the fingers of the manipulator hand, the position of the hand can only be fully determined if the posture of the hand is known. Assuming that the manipulator has a sufficient number of joints, the manipulator can be in any position and the position of the points between the fingers remains constant. To describe the posture of an object, we will fix a coordinate system on the object and give the representation of this coordinate system with respect to the reference system. In Figure 2-2, the coordinate system {B} is known to be fixed to the object in some way. The description in {B} relative to {A} is sufficient to indicate the attitude of object (A).

Coordinate System

A reference frame can be described in terms of the relation of one coordinate system with respect to another. The reference frame includes the concepts of position and posture, which is considered to be a combination of these two concepts in most cases. The position can be represented by a frame of reference in which the rotation matrix is the identity matrix and the position vector in this frame of reference determines the position of the described point. Similarly, if the position vector in the frame of reference is the zero vector, then it represents the posture.

4 DH Parameters

Definition

For rotational joint n , set $d=0.0$, the direction of X axis is the same as that of X_{n-1} axis, the origin position of coordinate system $\{n\}$ is selected to satisfy $d=0.0$. For prismatic joint n , the direction of axis z is set to meet $\theta=0.0$. When $d=0.0$, the origin of the coordinate system $\{n\}$ is selected to be located at the intersection of axis X_{n-1} and joint axis n .

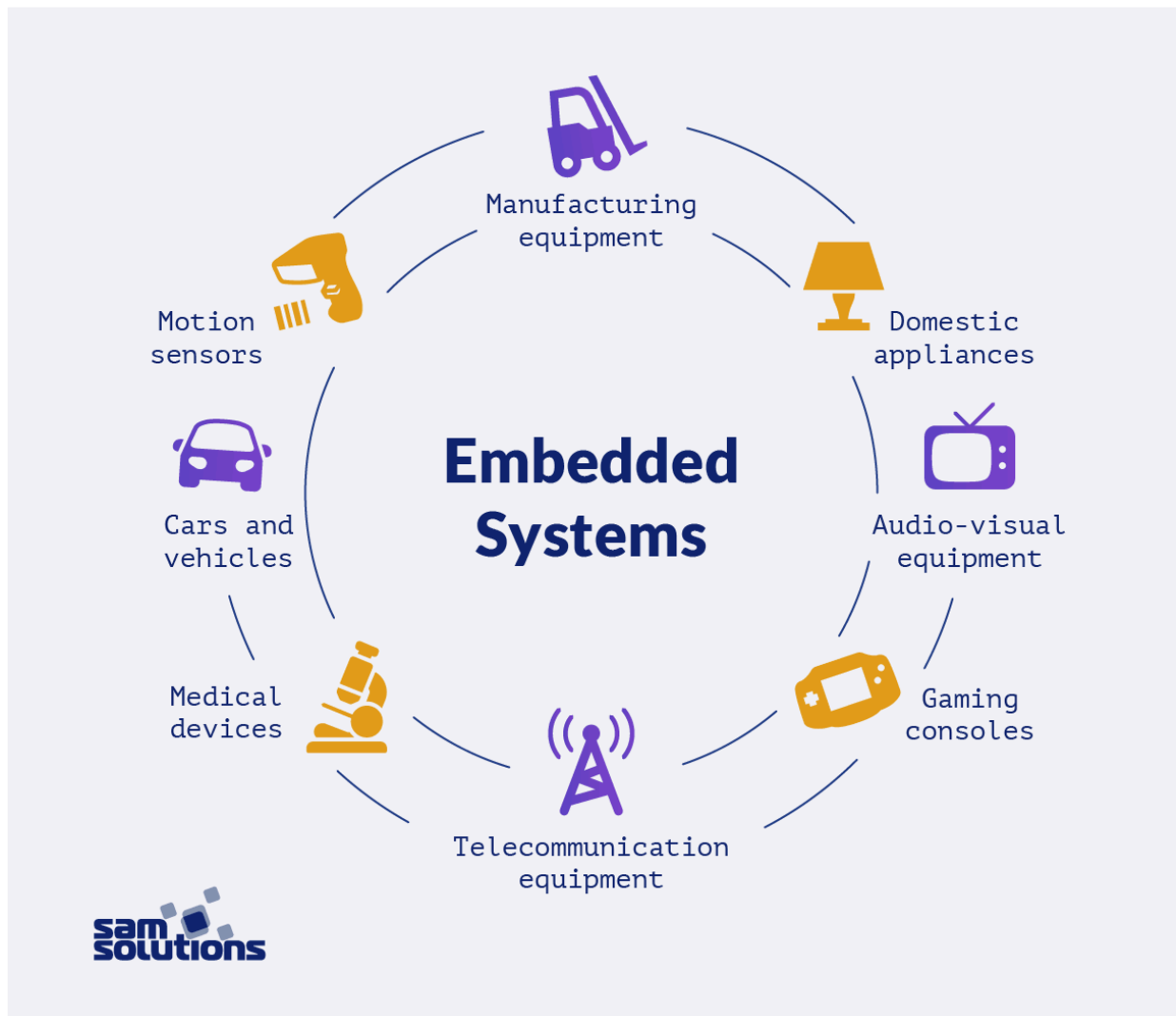
In the link coordinate system, if the link coordinate system is fixedly attached to the link as described above, the link parameters can be defined as follows:

- a_{i-1} : along x_{i-1} : move from the distance of z_{i-1} to z_i
- α_{i-1} : around x_{i-1} : rotate from the Angle of z_{i-1} to z_i
- d_i : along z_i : move from the distance of x_{i-1} to x_i
- θ_i : around z_i : rotate from the Angle of x_{i-1} to x_i

myCobot DH parameter

Joint	alpha	a	d	theta	offset
1	0	0	131.56	theta_1	0
2	PI/2	0	0	theta_2	-PI/2
3	0	-110.4	0	theta_3	0
4	0	-96	64.62	theta_4	-PI/2
5	PI/2	0	73.18	theta_5	PI/2
6	-PI/2	0	48.6	theta_6	0

Electronics



Introduction

Single-Chip Microcomputer (Microcontrollers/SCM) is a kind of integrated circuit chip, which uses VLSI technology to integrate the central processing unit CPU with data processing capabilities, random access memory RAM, read-only memory ROM, various I/O ports, interrupt systems, and timer /Counter and other functions (may also include display drive circuit, pulse width modulation circuit, analog multiplexer, A/D converter and other circuits) integrated into a silicon chip to form a small and complete microcomputer system, widely used in the field of industrial control. From the 1980s, it developed from the 4-bit, 8-bit microcontroller to the current 300M high-speed microcontroller.

Basic Structure

- **Arithmetic Unit**

Arithmetic Unit is composed of arithmetic logic unit (ALU), accumulator and register. The function of ALU is to perform arithmetic or logical operations on the incoming data. The input comes from two 8-bit data sources, one from the accumulator and the other from the data register. ALU can perform various operations on the data, such as Addition, Subtraction, AND, OR, Comparison, etc., and finally store the results in the accumulator.

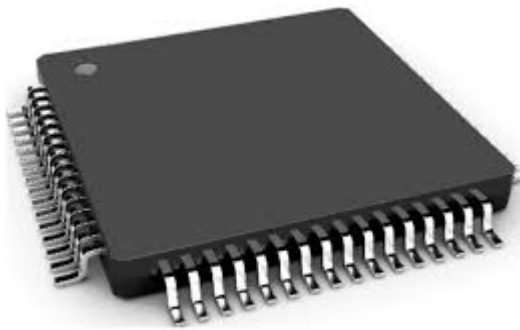
The arithmetic machine has two functions:

- Perform various arithmetic operations.
- Perform various logical operations and perform logical tests, such as a zero-value test or a comparison of two values.
- All operations performed by arithmetic unit are directed by the control signal issued by the controller, and an arithmetic operation produces an operation, and a logical operation produces a decision.
- **Controller**

Controller is composed of program counter, instruction register, instruction decoder, timing generator and operation controller, etc. It is the "decision-making body" of issuing commands, namely coordinating and directing the operation of the entire microcomputer system. Its main functions are :

- Extract an instruction from memory and indicates the location of the next instruction in memory.
- Decode and test the instructions, then generate the corresponding operation control signal to facilitate the execution of the specified actions.
- Direct and control the direction of data flow between CPU, memory, and input/output devices.

Microprocessor interconnects ALU, counter, register and control parts through an internal bus, and connects the external memory, input and output interface circuit through an external bus. External bus, also known as system bus, is divided into data bus DB, address bus AB and control bus CB. It can be connected with various peripheral devices through the input and output interface circuit.



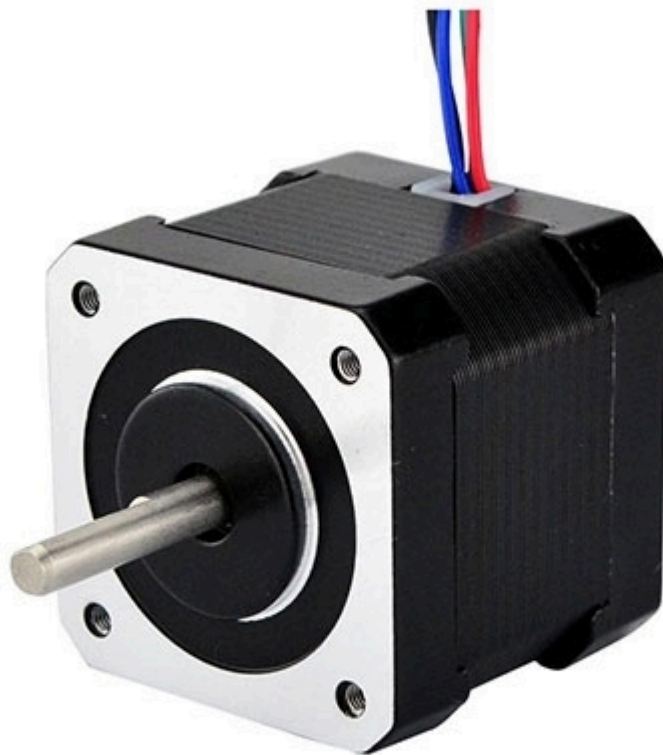
Motor & Steering Gear



Motor

According to the type of working power supply, it can be divided into:

- 1). DC(Direct Current) motor
 - 1.1). Brushless DC motor
 - 1.2). Brushed DC motor
 - 1.2.1). Permanent magnet DC motor
 - 1.2.1.1). Rare Earth permanent magnet DC motor
 - 1.2.1.2). DC Ferrite permanent magnet DC motor
 - 1.2.1.3). Aluminium Nickel-cobalt permanent magnet DC motor
 - 1.2.2). Electromagnetic DC motor
 - 1.2.2.1). DC series motor
 - 1.2.2.2). Shunt DC motor
 - 1.2.2.3). Separately Excited DC motor
- 2). AC (Alternating Current) motor
 - 2.1). Single-phase motor
 - 2.2). Three-phase motor



According to the application, it can be divided into:

- **Drive motor:** Electric tools (including drilling, buffing, polishing, grooving, cutting, reaming and other tools) motor, household appliances(including washing machine, electric fan, refrigerator, air conditioner, tape recorder, video recorder, DCD, vacuum cleaner, camera, hair dryer, electric shaver, etc.) motor, and other general small mechanical equipment (including all kinds of small machine tools, small machinery, medical equipment, electronic instruments, etc.) motor
- **Control motor:**
 - Stepping motor
 - Servo motor

What is Servo Motor



Servo motor (servo motor) refers to the engine that controls the operation of mechanical components in the servo system, and is an auxiliary motor indirect transmission device. The servo motor can control the speed and position accuracy very accurately, and can convert the voltage signal into torque and speed to drive the control object. The rotor speed of the servo motor is controlled by the input signal and can respond quickly. In the automatic control system, it is used as an executive element, and has the characteristics of small electromechanical time constant and high linearity. It can convert the received electrical signal into the motor shaft. angular displacement or angular velocity output. It is divided into two categories: DC and AC servo motors. Its main feature is that when the signal voltage is zero, there is no rotation phenomenon, and the speed decreases uniformly with the increase of torque. Servo Motor refers to the engine that controls the operation of mechanical components in the servo system, it is a kind of indirect speed change device for auxiliary motor.

How Servo Motors Work

1. The servo system is an automatic control system that enables the output controlled quantities such as the position, orientation, and state of the object to follow any changes in the input target (or given value). The servo mainly relies on pulses for positioning. Basically, it can be understood in this way that when the servo motor receives one pulse, it will rotate the angle corresponding to one pulse, thereby realizing the displacement. Because the servo motor itself has the function of sending out pulses, so every time the servo motor Rotating an angle, a corresponding number of pulses will be sent out, which echoes the pulses received by the servo motor, or is called a closed loop. In this way, the system will know how many pulses are sent to the servo motor and how many pulses are received at the same time. , in this way, the rotation of the motor can be precisely controlled, so as to achieve precise positioning, which can reach 0.001mm. DC servo motors are divided into brushed and brushless motors. Brushed motors have low cost, simple structure, large starting torque, wide speed regulation range, easy control, and require maintenance, but maintenance is inconvenient (carbon brush replacement), generates electromagnetic interference, and has environmental requirements. Therefore, it can be used in general industrial and civil occasions that are sensitive to cost. The brushless motor is small in size, light in weight, large in output, fast in response, high in speed, small in inertia, smooth in rotation and stable in torque. The control is complex, and it is easy to realize intelligentization. Its electronic commutation mode is

flexible, and it can be square wave commutation or sine wave commutation. The motor is maintenance-free, has high efficiency, low operating temperature, small electromagnetic radiation, long life, and can be used in various environments.

2. The AC servo motor is also a brushless motor, which is divided into synchronous and asynchronous motors. Synchronous motors are generally used in motion control. It has a large power range and can achieve great power. High inertia, low maximum rotational speed, and decreases rapidly as power increases. Therefore, it is suitable for applications that operate smoothly at low speeds.
3. The rotor inside the servo motor is a permanent magnet. The U/V/W three-phase electricity controlled by the driver forms an electromagnetic field. The rotor rotates under the action of this magnetic field. At the same time, the encoder that comes with the motor feeds back the signal to the driver, and the driver according to the feedback value Compared with the target value, the angle of rotation of the rotor is adjusted. The accuracy of the servo motor is determined by the accuracy (number of lines) of the encoder. The difference between AC servo motor and brushless DC servo motor in function: AC servo is better, because it is controlled by sine wave, and the torque ripple is small. DC servos are trapezoidal waves. But DC servos are simpler and cheaper.

Comparison of characteristics of servo motors

DC brushless servo motor features small moment of inertia, low starting voltage and low no-load current; abandoning the contact commutation system, greatly increasing the motor speed, the maximum speed is up to 100 000rpm; the brushless servo motor does not need an encoder when performing servo control It can also realize the control of speed, position, torque, etc.; there is no brush wear, and in addition to high speed, it also has the characteristics of long life, low noise, and no electromagnetic interference. DC brush servo motor features

4. Small size, quick action, quick response, large overload capacity, wide speed regulation range
5. Large torque at low speed, small fluctuation and stable operation
6. Low noise, high efficiency
7. The back-end encoder feedback (optional) constitutes the advantages of DC servo and so on
8. Large transformer range and adjustable frequency

The advantages of servo motors over ordinary motors

1. Accuracy: closed-loop control of position, speed and torque is realized; the problem of stepper motor out-of-step is overcome;
2. Speed: good high-speed performance, generally rated speed can reach 2000 ~ 3000 rpm;
3. Adaptability: strong anti-overload capability, able to withstand loads three times the rated torque, especially suitable for occasions with instantaneous load fluctuations and fast start requirements;
4. Stable: The low-speed operation is stable, and the stepping operation phenomenon similar to the stepping motor will not occur during low-speed operation. Applicable to occasions with high-speed response requirements;
5. Timeliness: The dynamic response time of motor acceleration and deceleration is short, generally within tens of milliseconds;
6. Comfort: heat and noise are significantly reduced. To put it simply: the ordinary motor that is usually seen will rotate for a while due to its own inertia after the power is turned off, and then stop. The servo motor and the stepper motor are to stop when they say they stop, and they can go when they say they are going, and the response is extremely fast. But the stepper motor has out-of-step phenomenon.

Application Scenarios of Servo Motors

There are too many applications for servo motors. As long as there is a power source, and there is a requirement for accuracy, it may generally involve a servo motor. Such as machine tools, printing equipment, packaging equipment, textile equipment, laser processing equipment, robots, automated production lines and other equipment that require relatively high process accuracy, processing efficiency and work reliability.

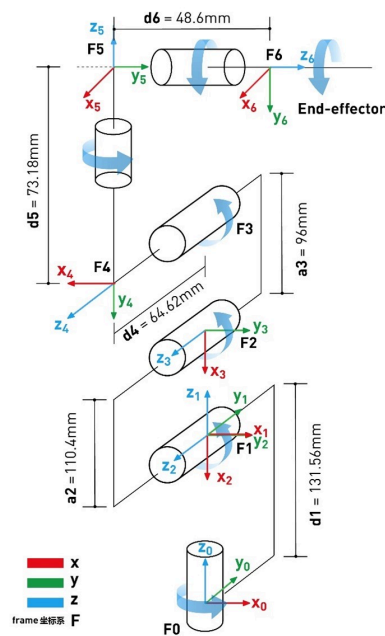
myCobot 280 algorithm

1 Structural Parameters

1.1 DH Parameters of Robotic Arm

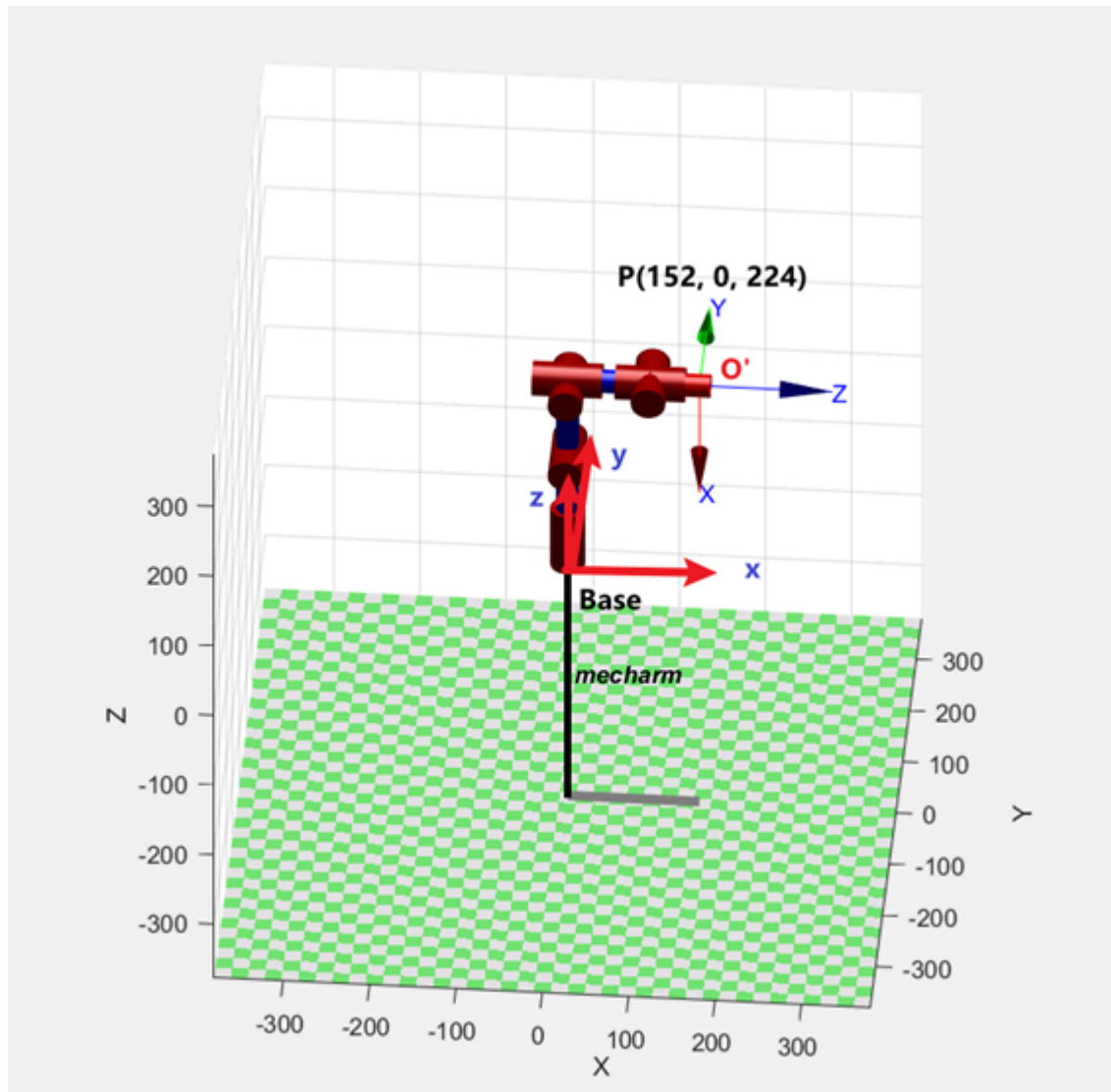
joint	theta	d	a	alpha	offset
1	q1	131.22	0	1.5708	0
2	q2	0	-110.4	0	-1.5708
3	q3	0	-96	0	0
4	q4	63.4	0	1.5708	-1.5708
5	q5	75.05	0	-1.5708	1.5708
6	q6	45.6	0	0	0

1.2 Kinematic Model



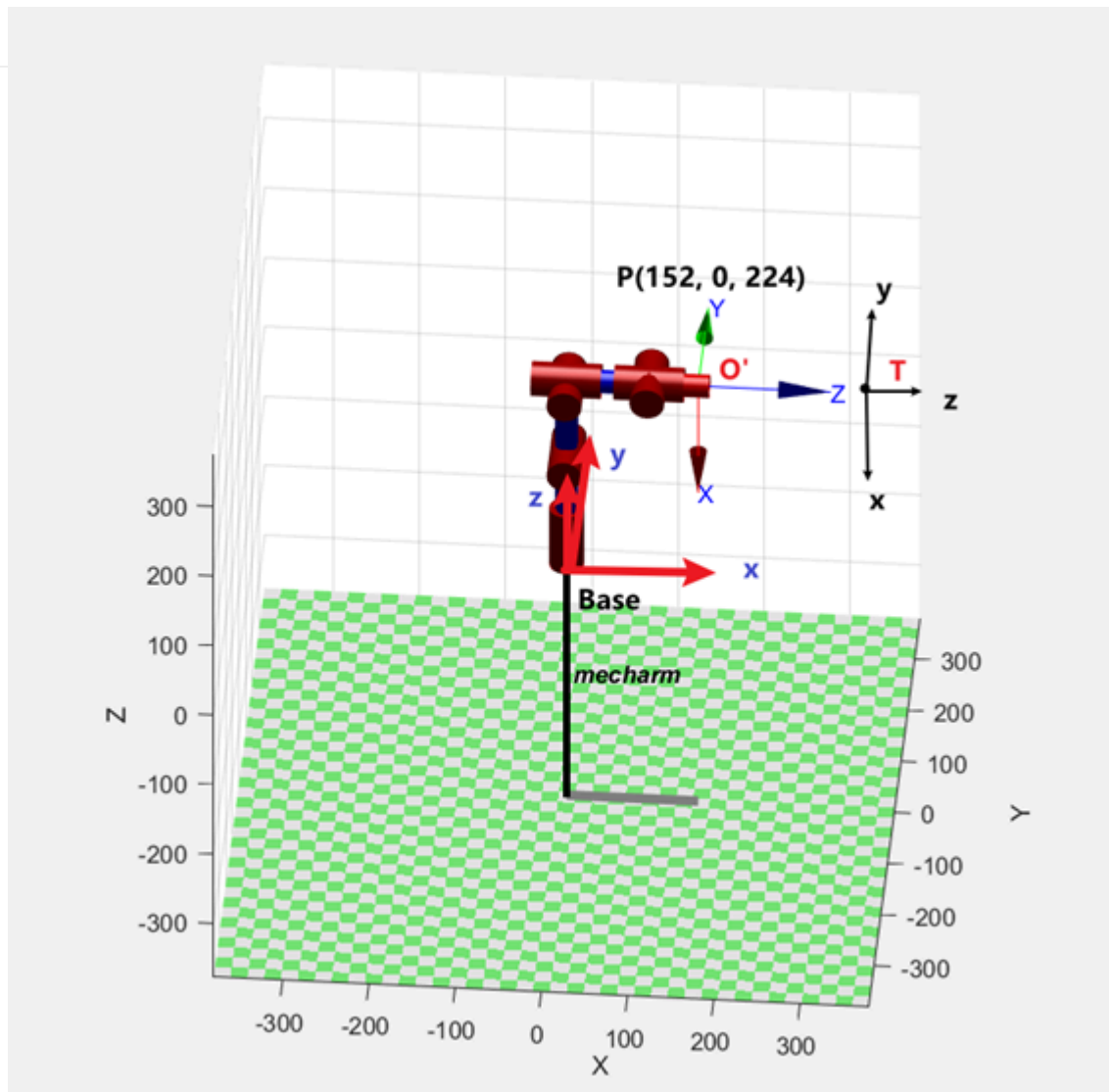
2 Coordinate System Introduction

2.1 Tool Coordinate System



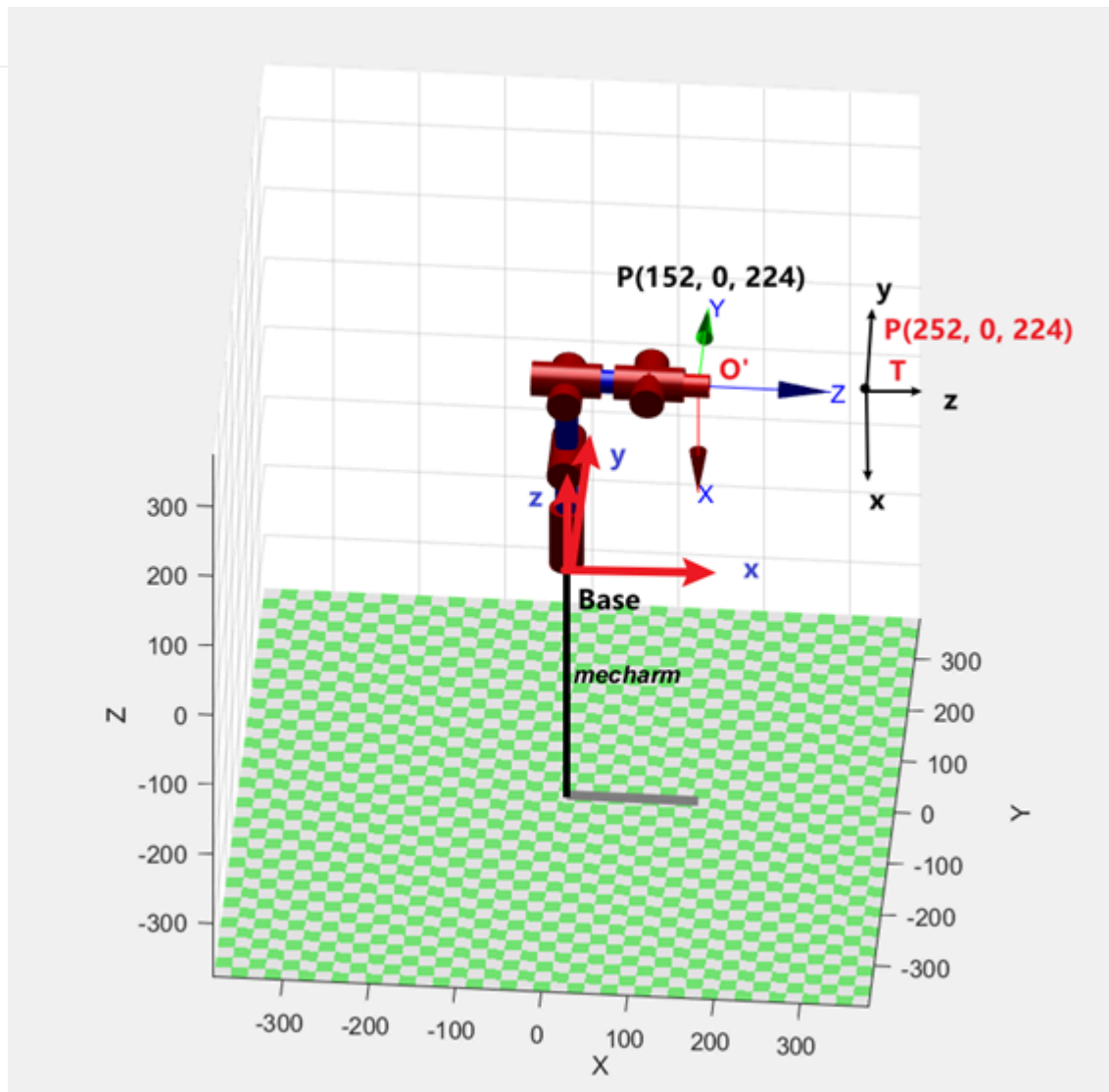
The figure shows the robot model of Mecharm270. Base in the figure represents the base coordinate system of the robot, O' represents the end flange coordinate system, and point P represents the position of the end of the manipulator relative to the base coordinate system ($x=152$, $y=0$, $z=224$)

Extend a certain pose on the basis of the end flange, and regard the set tool point as the end of the machine:



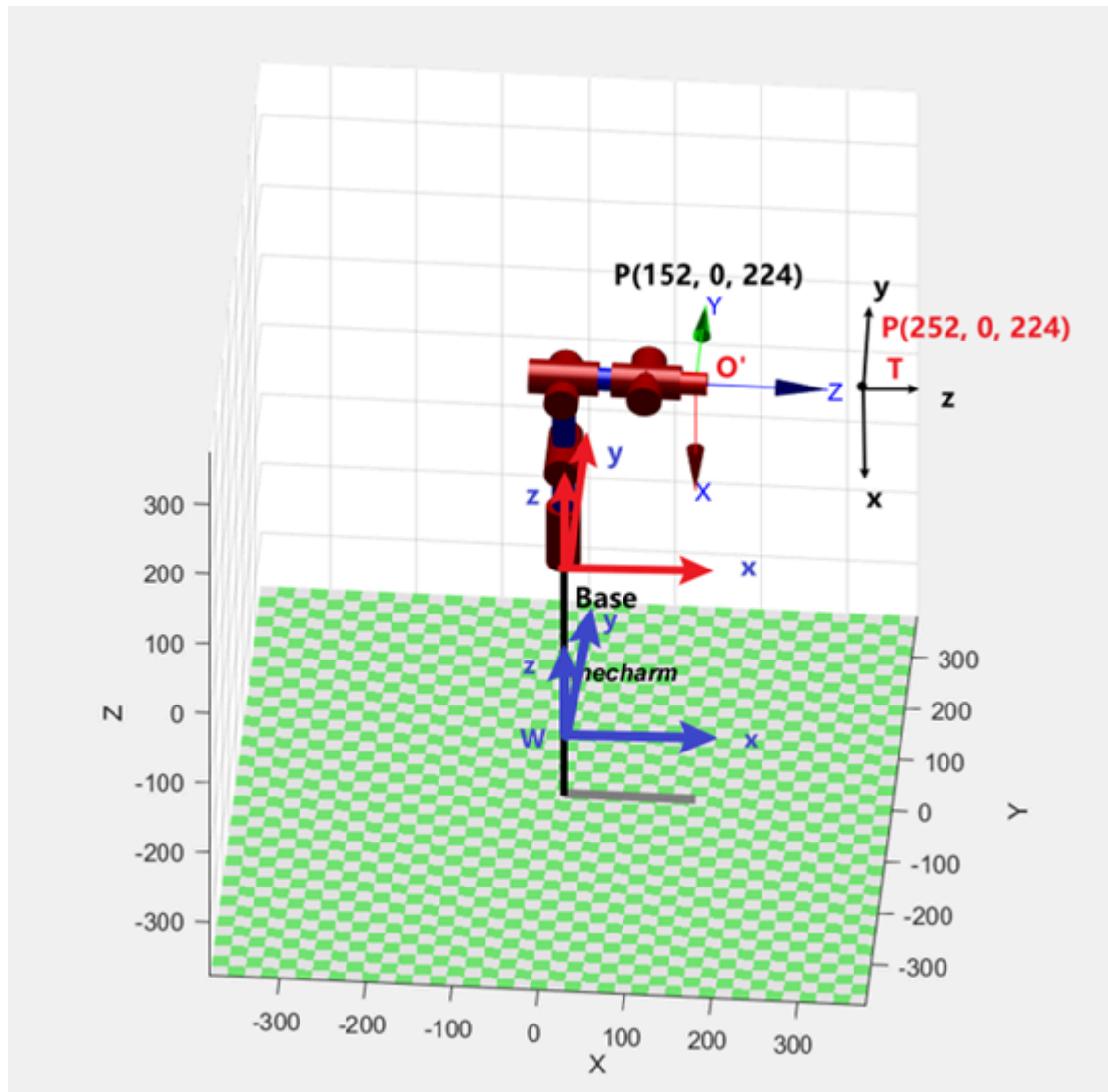
T in the figure is the set tool coordinate system. The posture of this coordinate system is consistent with O', and the relative displacement of the origin has occurred. Use the python function to set the tool coordinate system:

- `set_tool_reference([x, y, z, rx, ry, rz])` //Set tool coordinate system
- `set_end_type(1)` //Set the end coordinate system type as tool
- Assume that the tool coordinate system T is not rotated relative to O' ($rx = ry = rz = 0$)
- Assume that the origin of the tool coordinate system T is in the coordinate system O' at ($x = 0, y = 0, z = 100\text{mm}$)
- The final tool coordinate system parameter is `set_tool_reference(0, 0, 100, 0, 0, 0)`



Since the tool coordinate system is set, the end of the robot extends from O' to T at this time, and the coordinates of the end of the machine read at this time become $(152+100, 0, 224)$, and the coordinate posture movement will revolve around the tool point T rotate.

2.2 World Coordinate System



Section 2 introduces that by setting the tool coordinate system, the end coordinate system of the manipulator can be extended to a certain pose. We can also extend a certain pose on the basis of the base coordinate system of the manipulator by setting the world coordinate system. The set world coordinate system will replace the original Base coordinate system and become the new base coordinate system.

W in the figure is the set world coordinate system. The posture of this coordinate system is consistent with Base, and the relative displacement of the origin has occurred. Use the python function to set the world coordinate system:

- `set_world_reference([x, y, z, rx, ry, rz])` //Set the world coordinate system
- `set_reference_frame(1)` //Set the base coordinate system type to the world
- Assuming that the world coordinate system W has not rotated relative to Base($rx = ry = rz = 0$)
- Suppose the origin of the world coordinate system W is in the coordinate system Base ($x = 0, y = 0, z = -100\text{mm}$)
- The final world coordinate system parameter is `set_world_reference(0, 0, -100, 0, 0, 0)`

Since the world coordinate system is set, the origin of the robot extends from Base to W at this time, and the O' coordinate read at this time becomes (152, 0, 224+100).

Application of Basic Functions

1 myStudio

Before using the equipment, you have to **install the driver and update the firmware**.

1.1 Installing the driver

According to the operating system used by him, the user can click the button below to download the zip file of the corresponding **CP210X** or **CP34X** drivers. After decompressing the file, select and install the installation package corresponding to the operating system.

There are two kinds of driver chip versions: **CP210X** (suitable for CP2104 version)/**CP34X** (suitable for CH9102 version) driver zip files. If you are not sure of the USB chip used for your equipment, you can install two drivers at the same time. (During the installation of **CH9102_VCP_SER_MacOS**, an error may be reported; however, the installation has been done, and the error can be ignored.)

For Mac OS, ensure correct settings of the system "Preferred settings -> Security and privacy -> General" before installation, and allow the user to get it from App Store or an approved developer.

- Download the **Basic** serial port driver **CP210X**

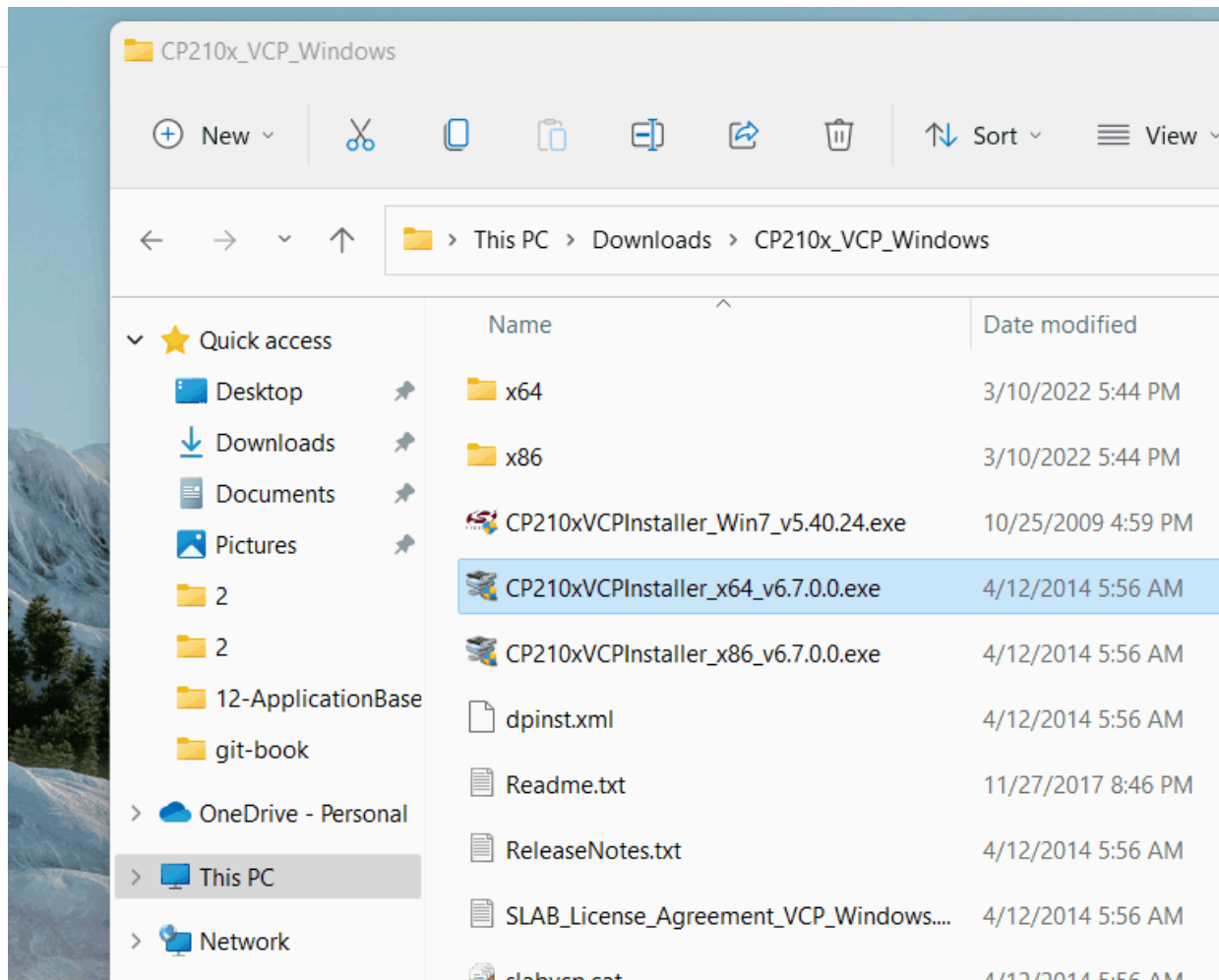
CP210X

- [Windows10](#)
- [MacOS](#)
- [Linux](#)

CP34X

- [Windows10](#)
- [MacOS](#)

- Download the **Atom** serial port driver for the end.
 - [Windows10](#)



1.2 Updating equipment firmware

Prior to development, the user is required to confirm whether his equipment firmware is the latest version so that he can use the equipment better during subsequent development.

The user can update the firmware through **myStudio**.

2 Factory firmware introduction

2.1 Drag teaching

Robot drag teaching is a process during which the operator can drag the joints of the robot directly to make them do ideal postures and then make records corresponding thereto. The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.

2.2 Robot arm calibration

Calibrating the robot arm is the precondition for precise control of the robot arm, and setting joint zero and initializing the potential of the motor are basic jobs for subsequent advanced development.

2.3 Communication forwarding

Communication timeliness is vital to the micro-controller robot arm. For such arm, we often send control instructions to **M5Stack-basic** at the bottom. Through communication forwarding, the end effector analyzes the instructions and then implements target actions. At present, micro-controller devices have three methods of communication: **serial port, Bluetooth, and WIFI**. The user may choose an applicable method of communication for programming.

2.4 Connection detection

Link test is a detection function that uses the motor in the robot arm and the connection state of **Atom**. The function allows the user to remove equipment faults easily.

During the link test, the connection state of the equipment for the robot arm, including the **connection of the servo** and the **communication state of Atom** can be seen. In micro-controller devices, the versions of their current firmwares are shown on M5Stack-basic.

3 Use for the first time

After knowing the current function of the firmware, connect and fix the machine by following the steps in this chapter, and start applying the basic functions of the equipment.

Safety Instructions

1 Synopsis

This chapter details general safety information for personnel performing installation, maintenance, and repair work on elephant robots. Read and understand the contents and precautions in this chapter before carrying, installing, and using it.

2 Hazard identification

The safety of cooperative robots is based on the proper configuration and use of robots. Furthermore, injury or damage caused by the operator may occur even if all safety instructions are followed. Therefore, it is very important to understand the safety risks of robot use in order to prevent them.

Table 1-1 to 3 lists the common security risks that may occur when robots are used:

Table 1-1 Risk level Security risks

 Hazard
1 Personal injury or robot damage caused by improper handling of the robot.
2 If the robot is not fixed as required, for example, the screw is missing or the screw is not tight, or the locking capacity of the base is insufficient to support the robot to move at high speed, the robot will fall over, resulting in personal injury or robot damage.
3 The robot's safety function fails to play its role due to the incorrect configuration of safety functions or the lack of safety protection tools.

Table 1-2 Warning security risks

Warning
1 Do not stay within the moving range of the robot when debugging the program. Improper safety configurations may not prevent collisions that may cause personal injury.
2 The robot's connection to other devices may lead to new hazards, requiring a thorough re-assessment of the risk.
3 Scratches and punctures are caused by sharp surfaces such as other equipment or robot end-effector in the working environment.
4 The robot is a precision machine. Stepping on the robot may cause damage.
5 If the clamping device is not in place, or the power supply of the robot is turned off, or the object is not removed before the air source (it is not determined whether the end-effector is firmly clamped because the object loses power), it may cause danger, such as damage to the end-effector and injuries to people.
6 The robot is at risk of accidental movement. Do not stand under any axis of the robot under any circumstances!
7 The robot is a precision machine, which may cause vibration and damage to the internal parts of the robot if it cannot be placed smoothly during handling.
8 The robot is a precision machine, which may cause vibration and damage to the internal parts of the robot if it cannot be placed smoothly during handling.

Table 1-3 Potential safety hazards that may lead to electric shock

Danger Electric Shock
1 Unknown hazards may arise when using non-original cables.
2 Electrical equipment contact with liquid may cause leakage hazard.
3 Electrical connection error may cause electric shock.
4 Be sure to switch off the power supply of the controller and related devices and remove the power plug before replacement. If the operation is carried out with power on, it may cause electric shock or failure.

3 Safety Precautions

The following safety rules should be followed when using the manipulator:

- The mechanical arm belongs to live equipment. Non-professionals are not allowed to change the circuit at will, otherwise it may cause damage to the equipment or human body.
- When operating mechanical arms, comply with local laws and regulations. The safety precautions and dangers, Warnings, and precautions described in this manual are only supplements to the local safety regulations.
- Please use the mechanical arm within the specified environment. Exceeding the specifications and load conditions of the mechanical arm will shorten the service life of the product and even damage the equipment.
- The person installing, operating and maintaining the myCobot arm, anyway, has to be rigorously trained on safety precautions and the right way to operate and maintain the robot.
- Anyway, don't use the product in humid environments for long periods of time. This product is a precision electronic component, which will damage the equipment in damp environment for a long time.
- Anyway, don't use the product in humid environments for long periods of time. This product is a precision electronic component, which will damage the equipment in damp environment for a long time.
- Highly corrosive cleaning is not suitable for cleaning the mechanical arm, and anodized parts are not suitable for immersion cleaning.
- Unconsciously, do not use the device without installing a base to avoid damaging the device or accidents, instead use the device in a fixed environment without obstacles.
- Do not use other power adapters for power supply. If the equipment is damaged due to the use of non-standard adapters, the after-sales service will not be included.
- Do not disassemble, disassemble, or unscrew the screws or shell of the manipulator. If disassembled, no warranty service can be provided.
- Personnel without professional training are not allowed to repair the faulty products and dismantle the mechanical arm without permission. If the products fail, please contact myCobot technical support engineers in time.
- If the product is discarded, please comply with the relevant laws to properly dispose of industrial waste and protect the environment.
- A child uses a device at some point, forcing someone to monitor the process and switch it off when it's finished.
- When the robot is moving, do not extend your hand into the movement range of the robot arm, for fear of collision.
- It is strictly prohibited to change, remove or modify the nameplate, description, icon and mark of the manipulator and related equipment.
- Please be careful in handling and installation. Put the robot gently according to the instructions on the packing case and place it correctly in the direction of the arrow. Otherwise, the machine may be damaged.
- **Do not burn other product drivers from Atom terminal, or burn firmware using unofficial recommendations. If the equipment is damaged due to the user burning other firmware, it will not be in the after-sales service.**

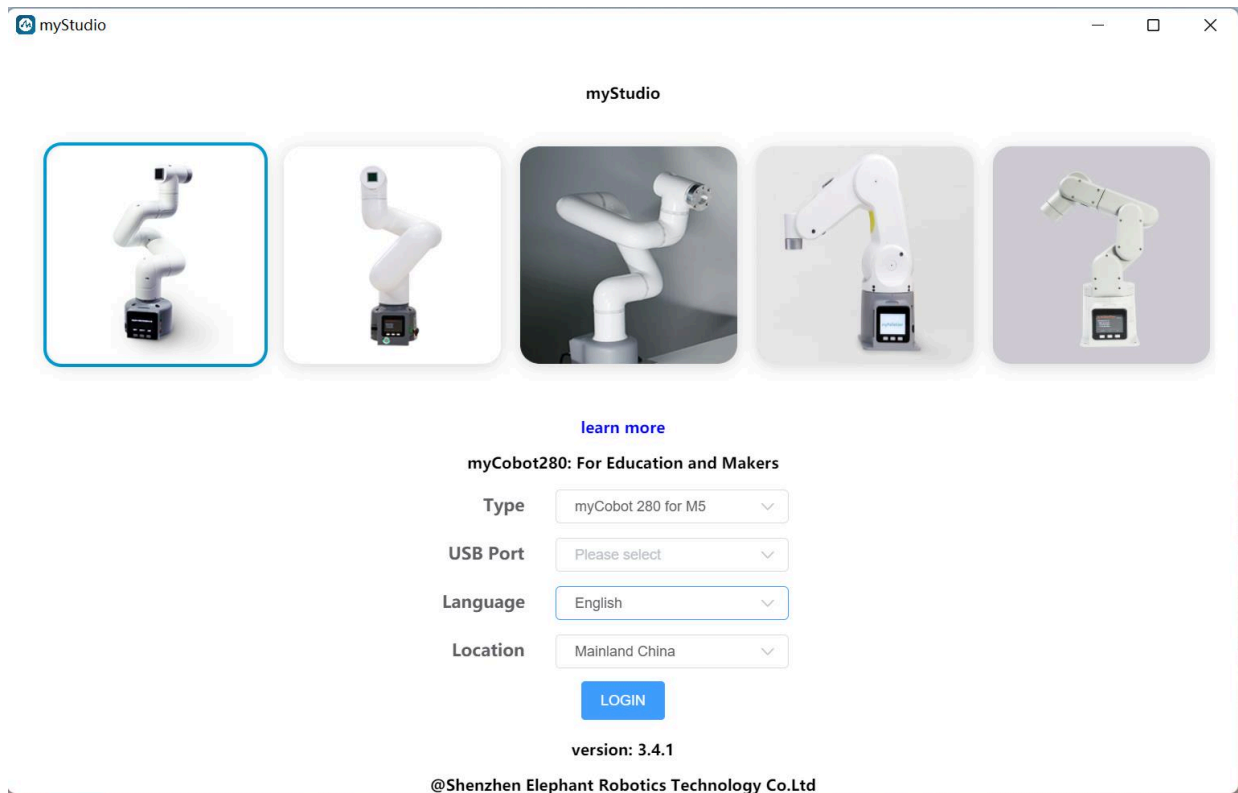
If you have any questions or suggestions about the contents of this manual, please log in the official website of Elephant Robot and submit relevant information :

<https://www.elephantrobotics.com>

Please do not use the mechanical arm for the following purposes :

- Cost of healthcare in life-critical applications.
- Buying a bus can cause an explosion in an environment.
- Lent is used directly without a risk assessment.
- Cost of using a security function at a low level.
- Lo-fi does not conform to the use of robot performance parameters.

myStudio



1 Reasons for Designing myStudio

- Because myStudio is a one-step application platform for a variety of robots.
- Because it is easy for users to select corresponding firmwares to meet demands and acquire related tutorial data online.

2 Latest Version and Supported Systems

- Version 3.4.1(updated on Oct 1, 2022)
- Supported Systems: Windows, Mac and Linux

3 Functions

- Burning and updating firmwares
- Providing tutorial data, such as user manuals and tutorial video
- Providing information about maintenance and repair

4 Matchable Robots

- myCobot 280
 - myCobot 280 M5

- myCobot 280 PI
 - myCobot 280 Jetson Nano
 - myCobot 280 for Arduino
- myCobot 320
 - myCobot 320 M5
 - myCobot 320 PI
- myPalletizer 260
 - myPalletizer 260 M5
 - myPalletizer 260 PI
- mechArm 270
 - mechArm 270-M5
 - mechArm 270 PI
- myCobot PRO 600
- myBuddy 280

5 Recommended Firmwares

The optimal firmwares varies with the type of robots in use. Recommended firmwares for different types of robots are listed as follows.

myCobot 280 series

myCobot 280 series have 4 versions: M5、PI、Arduino and Jetsonnano. With different core for signal connection, different firmwares are required to be burnt.

Robot Version	Core	Firmware to be Burnt	Recommended Firmware
M5	M5Stack-Basic	miniRobot	v2.1 is recommended for dragging teaching, wifi connection and bluetooth connection
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before, or not labelled; v5.1 is recommended for robots labelled ER28001202200416 and after
PI	RaspberryPI 4B	ubuntu	v18.04.is recommended
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before, or not labelled; v5.1 is recommended for robots labelled ER28001202200416 and after
Arduino	mega2560	transponder	v1.0 is recommended
	mkrwifi1010	transponder	v1.0 is recommended
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before, or not labelled; v5.1 is recommended for robots labelled ER28001202200416 and after
Jetson nano	JestonNano	ubuntu	v18.04.is recommended
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before, or not labelled; v5.1 is recommended for robots labelled ER28001202200416 and after

myCobot 320 (2022) series

myCobot 320 (2022) series have two versions: M5 and PI. With different core for signal connection, different firmwares are required to be burnt.

Robot Version	Core	Firmware to be Burnt	Recommended Firmware
M5	M5Stack-basic	miniRobot	v2.0 is recommended
	Atom	atomMain	v5.0 is recommended
	Pico	picoMain	v1.0 is recommended
PI	RaspberryPI 4B	ubuntu	v18.04. is recommended
	Atom	atomMain	v5.0 is recommended
	Pico	picoMain	v1.0 is recommended

****myCobot 320 (2020) series ****

Robot Version	Core	Firmware to be Burnt	Recommended Firmware
M5	M5Stack-basic	miniRobot	v1.0 is recommended
	Atom	atomMain	v4.2 is recommended
PI	RaspberryPI 4B	ubuntu	v18.04. is recommended
	Atom	atomMain	v4.2 is recommended

myPalletizer 260 series

myPalletizer 260 series have two versions: M5 and PI. With different core for signal connection, different firmwares are required to be burnt.

Robot Version	Core	Firmware to be Burnt	Recommended Firmware
M5	M5Stack-Basic	miniRobot	v1.1 is recommended
PI	Atom	atomMain	v1.1 is recommended

1 Environment Building

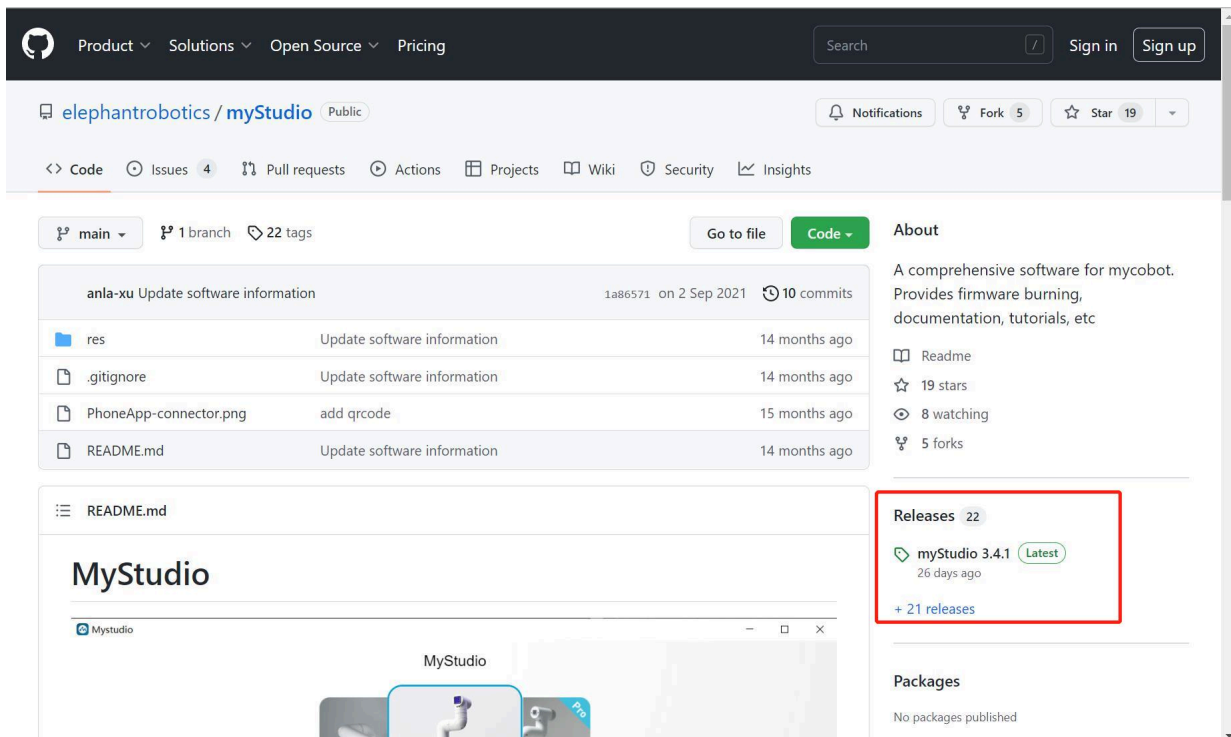
1.1 Download and Installation

Note: The installation path of myStudio installation cannot have any spaces

address for downloading:

1. GitHub

- Click on `myStudio` at the right side and download the version corresponding to your PC. **Do not install myStudio into files with space directory.**



- Different suffixes signifies versions applicable for different systems. -
 - *.tra.xz is applicable to Linux
 - *.dmg is applicable to Mac
 - *.exe is applicable to Window

▼ Assets

4

 [myStudio-3.4.1-arm64.ApplImage](#)

 [myStudio.Setup.3.4.1.exe](#)

 [Source code \(zip\)](#)

 [Source code \(tar.gz\)](#)

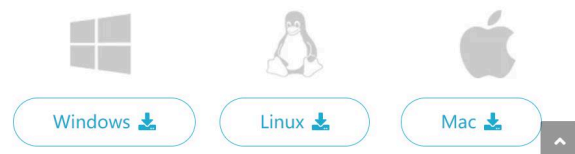
2. Elephant Robotics Official Website

Software Downloads

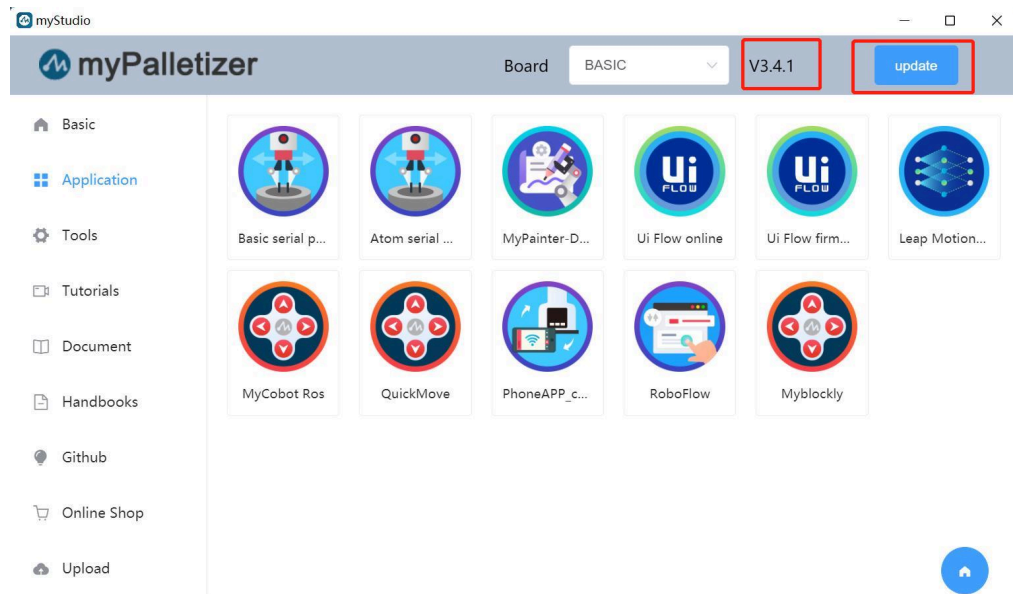


myStudio

myStudio is a one-stop platform for robots of myRobot/myCobot. The main functions of myStudio are: 1) Update the firmware; 2) Provide video tutorials on how to use the robot; 3) Provide maintenance and repair information (such as video tutorials, Q&A, etc.).



Notice: Download the latest version. You can log in to check the present version of myStudio and update to the latest.



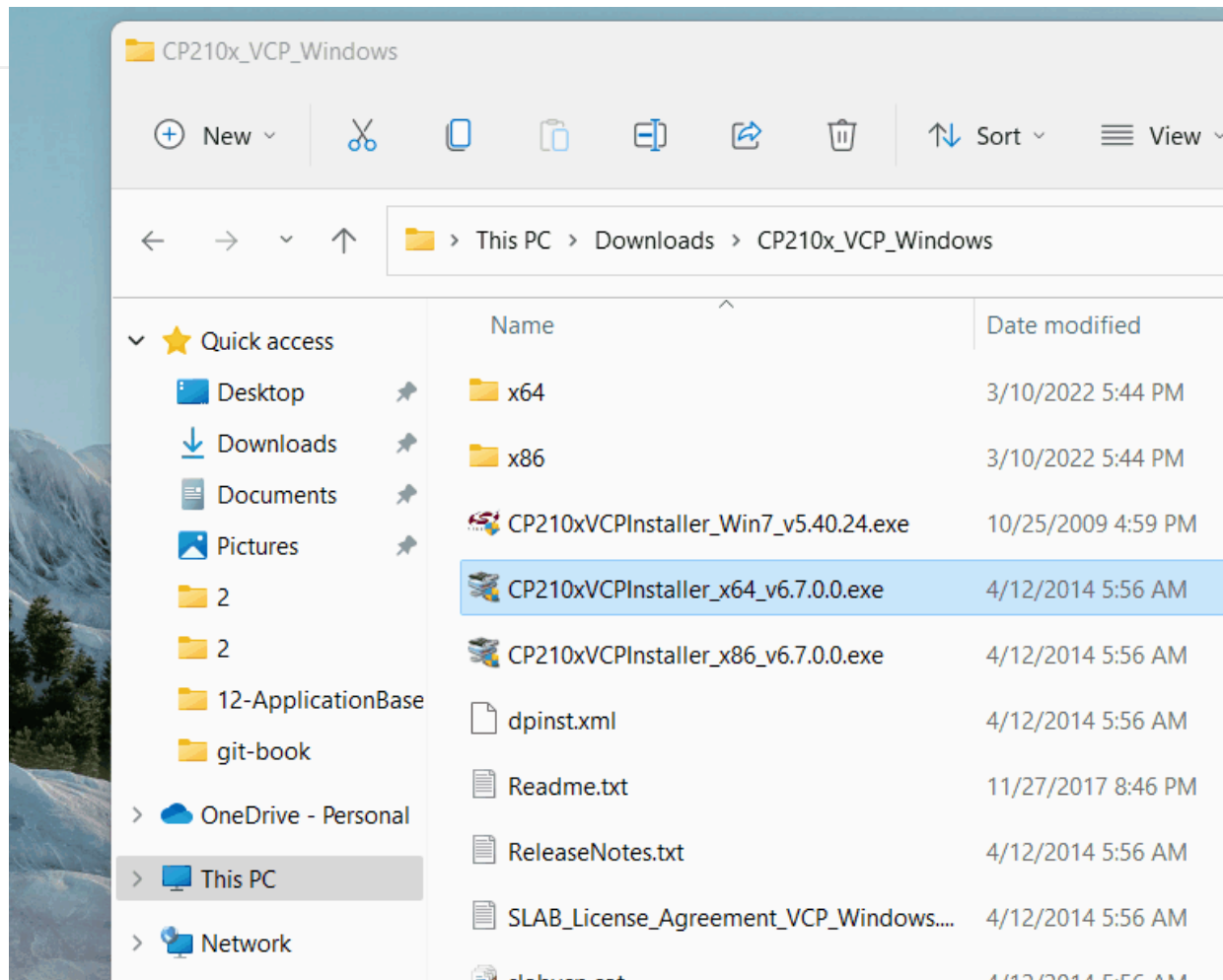
1.2 Serial Port Driver Installation

Download corresponding serial port driver according to the USB chip on your PC. CP210X is suitable for CP2104 version and CP34X is suitable for CH9102 version. You can install both of them if you are unable to confirm the type of USB chip. Go to the address below to download and install serial port driver.

- serial port driver for M5Stack Basic:
 - CP210X: [Windows 10](#), [MacOS](#), [Linux](#)
 - CP34X: [Windows 10](#), [MacOS](#)
- serial port driver for Atom:
 - [Windows 10](#)

Notice:

- An error may be reported during installation of CH9102_VCP_SER_MacOS. However, the installation already completes. Just ignore the error.
- Make sure that settings of the system \"Preferred settings -> Security and privacy -> General\" is correct and allow the user to get it from App Store or an approved developer.



1.3 How to distinguish between CP210X chip and CP34X chip

- As shown in the GIF below, go to `equipment manager` and check `ports (COM and LPT)` .



- If the ports (COM and LPT) show USB-Enhanced-SERIAL CH9102 , the chip is CP34X.



- If the ports (COM and LPT) show Silicon Labs CP210x USB to UART Bridge , the chip is CP210X.

- ▼  PROTEGO-7S2B7J5
 - >  USB 连接器管理器
 - >  安全设备
 - >  处理器
 - >  传感器
 - >  磁盘驱动器
 - >  存储控制器
 - >  打印队列
 - >  打印机
 - >  电池
 - ▼  端口 (COM 和 LPT)
 -  Silicon Labs CP210x USB to UART Bridge (COM6)
 - >  固件
 - >  计算机
 - >  监视器
 - >  键盘
 - >  蓝牙
 - >  人机接口设备
 - >  软件设备
 - >  软件组件
 - >  生物识别设备
 - >  声音、视频和游戏控制器
 - >  鼠标和其他指针设备
 - >  通用串行总线控制器
 - >  网络适配器
 - >  系统设备
 - >  显示适配器
 - >  音频输入和输出
 - >  照相机

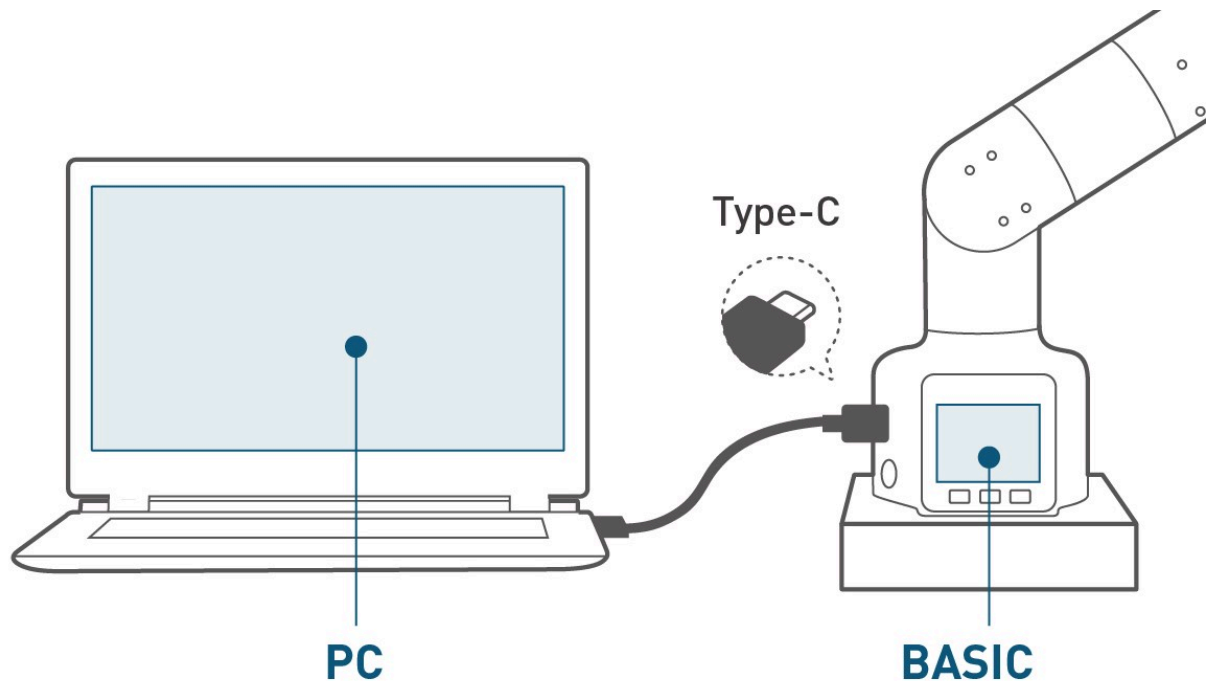
2 Burning and Updating Firmwares

Tutorial Video: <https://www.bilibili.com/video/BV1Qr4y1N7B5/>

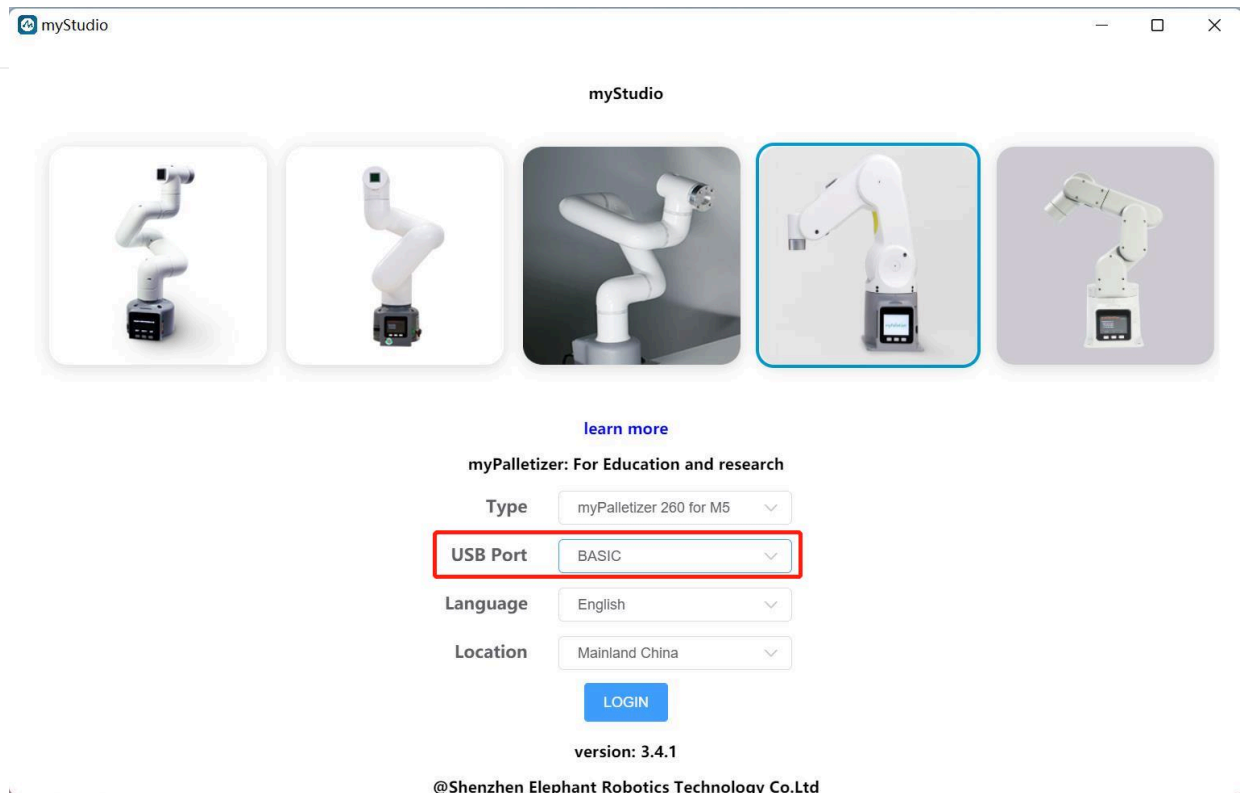
2.1 Burning M5Stack-basic Firmwares

Notice: PI version is not required to burn M5Stack-basic firmwares

Step 1: Connect M5Stack-basic with PC with USB.

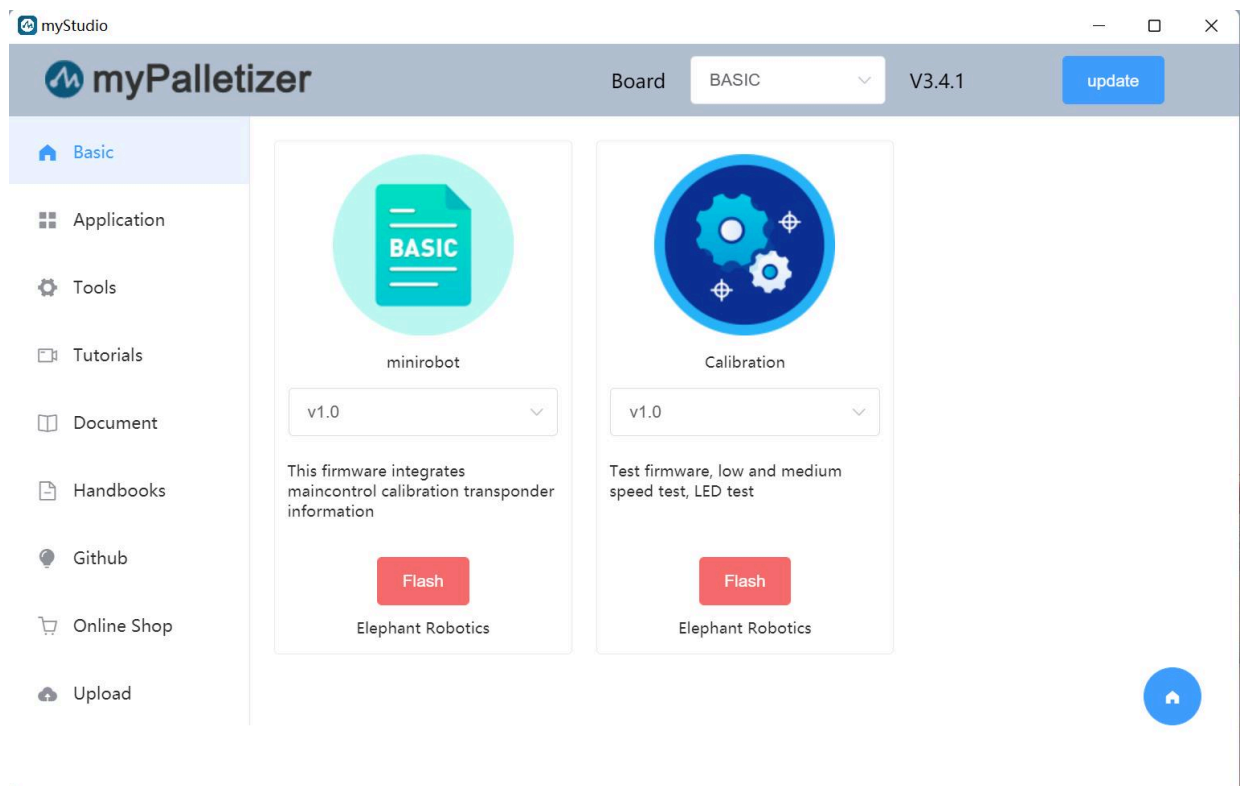


Step 2: Select `USB Port`. After connecting with PC, `USB Port` shows `M5Stack-basic`. The figure below takes myPalletizer 260 M5 as an example.



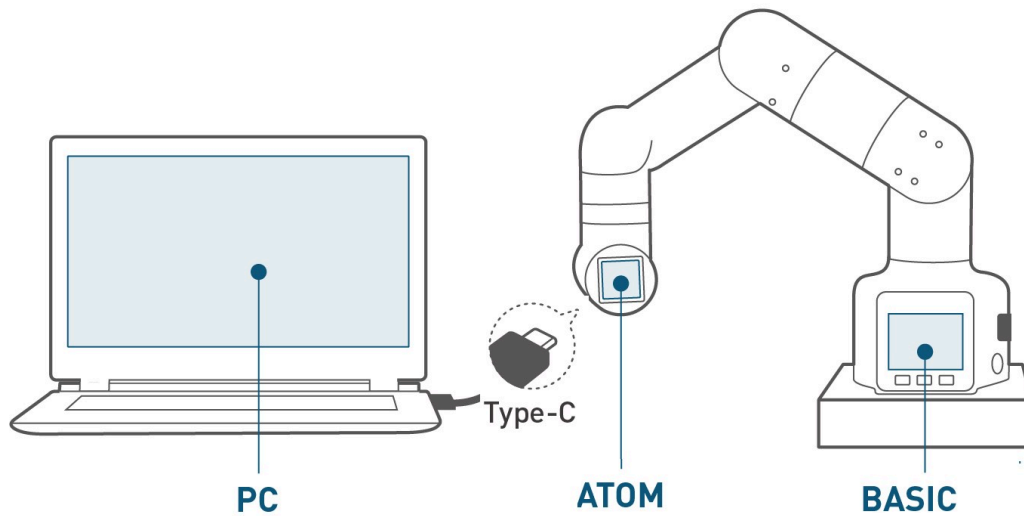
- Click on `LOGIN` -> `M5Stack-basic` , and then you can burn firmwares.

Notice: As 280 PI/Jetson nano/Arduino versions do not have M5Stack-basic, the `USB Port` cannot show anything after connection.

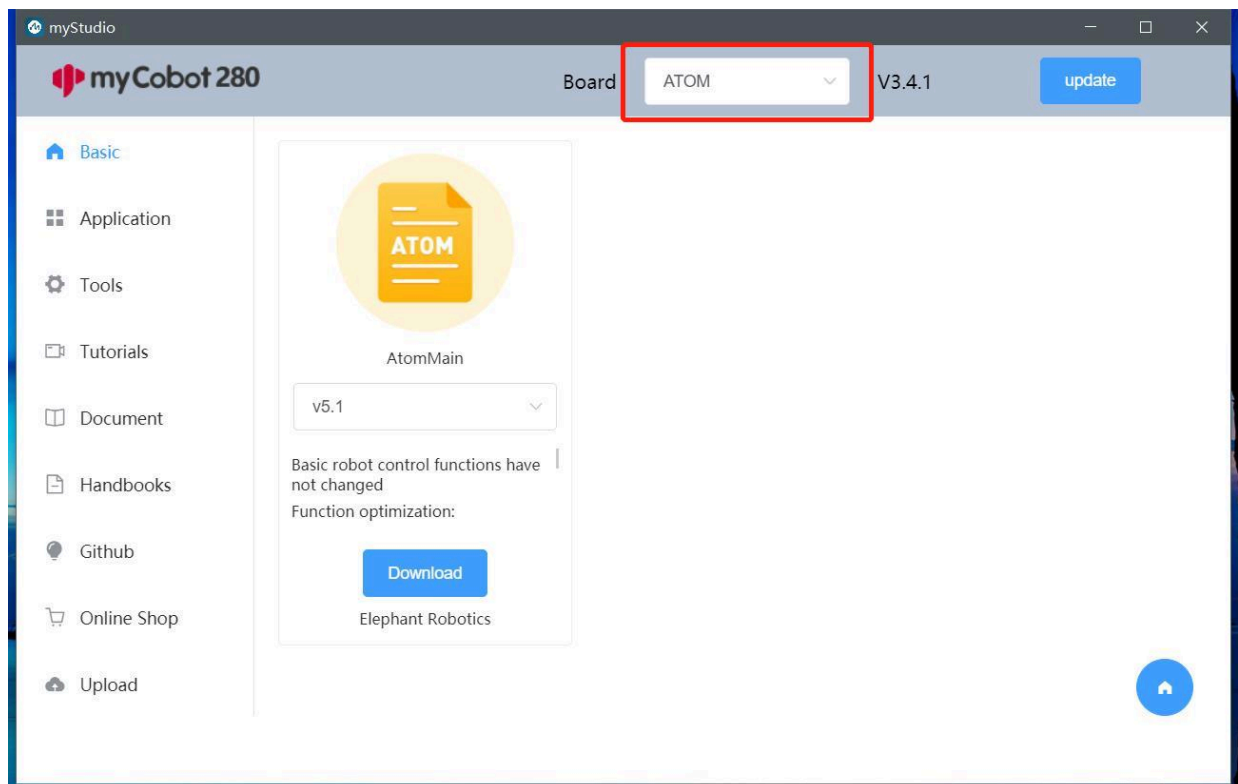


2.2 Burning Atom Firmware

Step 1: Connect Atom with PC with USB.



Step 2: Select `ATOM` at Board, and then burn AtomMain. The picture below takes myCobot 320 as an example.

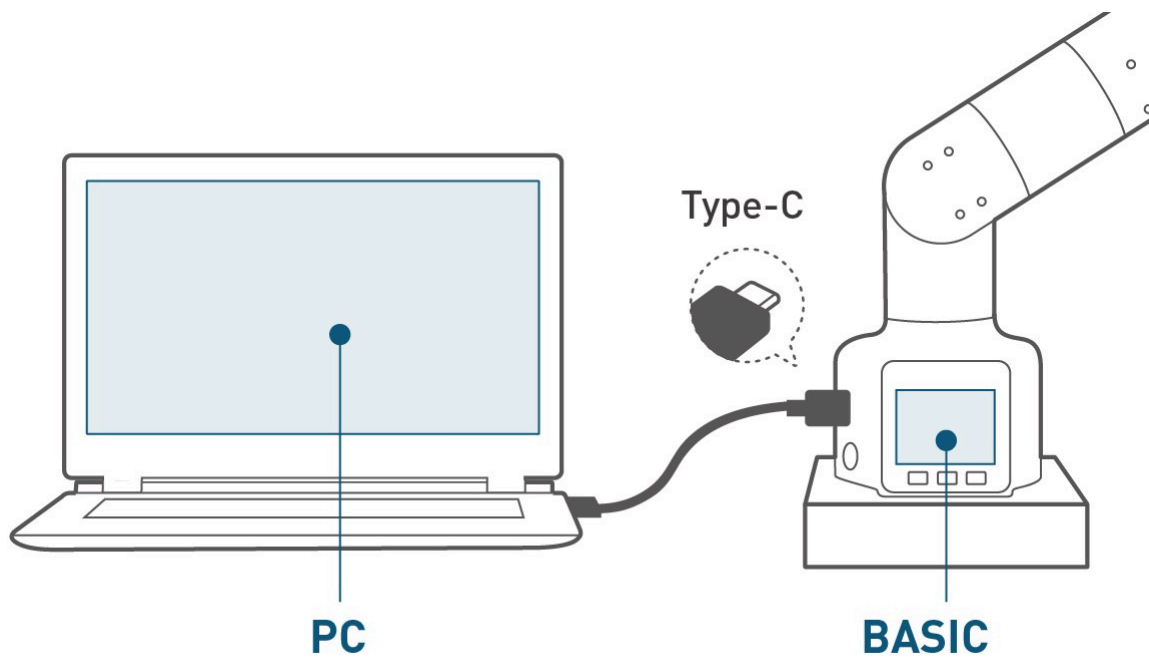


2.3 Firmware Burning for myCobot 320 series

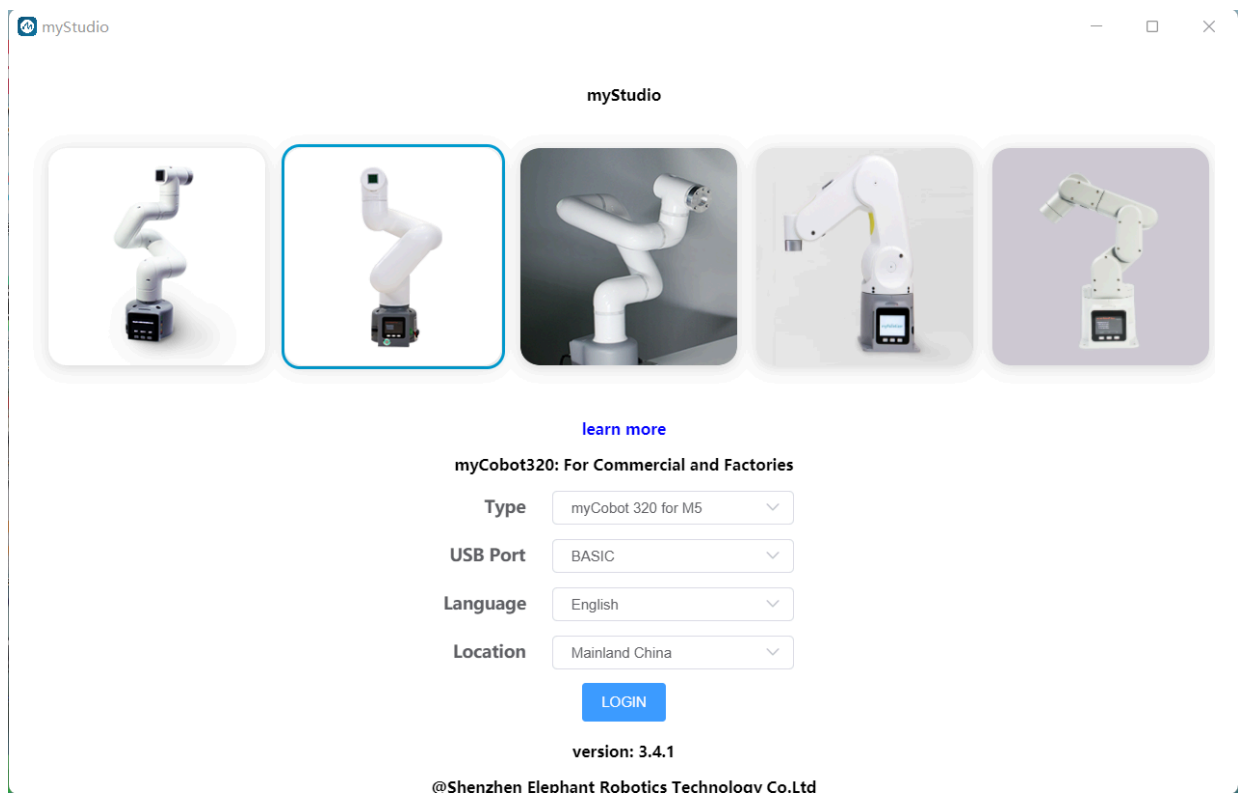
M5Stack-basic and picoMain are required to be burnt on myCobot 320 series. It should be noted that the two firmwares are burnt by different type-c interfaces.

1.Burning M5Stack-basic: miniRobot

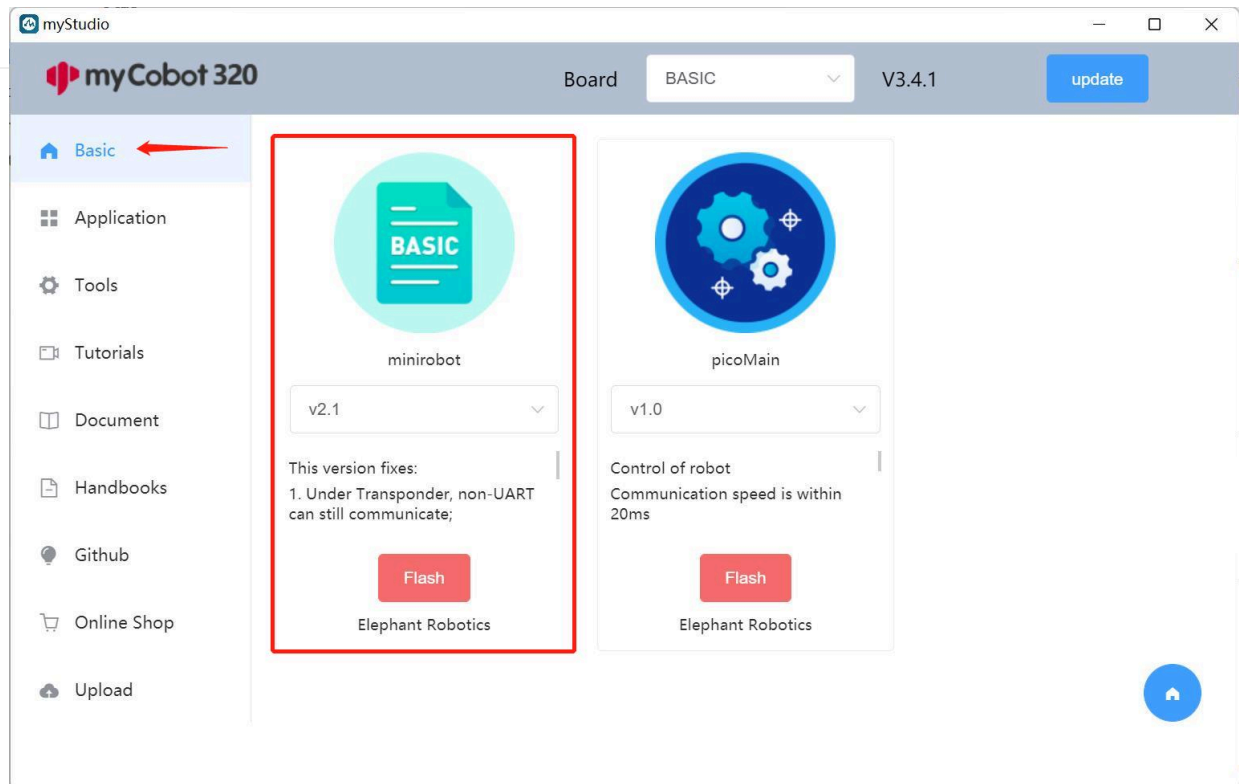
Step 1: Connect M5Stack-basic and PC.



Step 2: Select the 320 series, language and region, and then click on **LOGIN**.



Step 3: Click on **Basic**, and select minirobot to burn.



2. Burning Pico Firmware: picoMain

Step 1: Connect M5Stack-basic with PC.

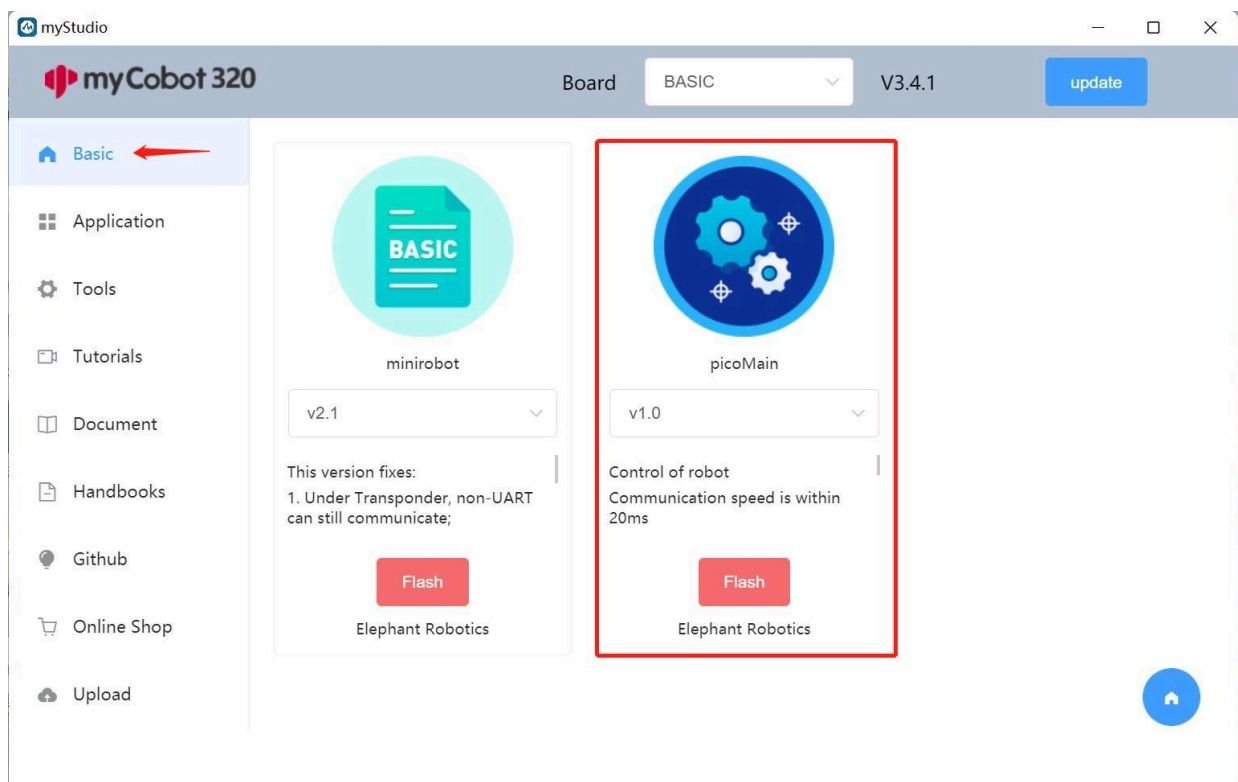


Step 2: Select **Flash** (press **UP** once or **DOWN** four times), and then the screen goes dark for approximately 30 seconds.



Step 3: select picoMain to burn via myStudio.

Notice: picoMain is required to be burnt during the sleep period of **Flash** mode. Otherwise it may fail to burn picoMain.



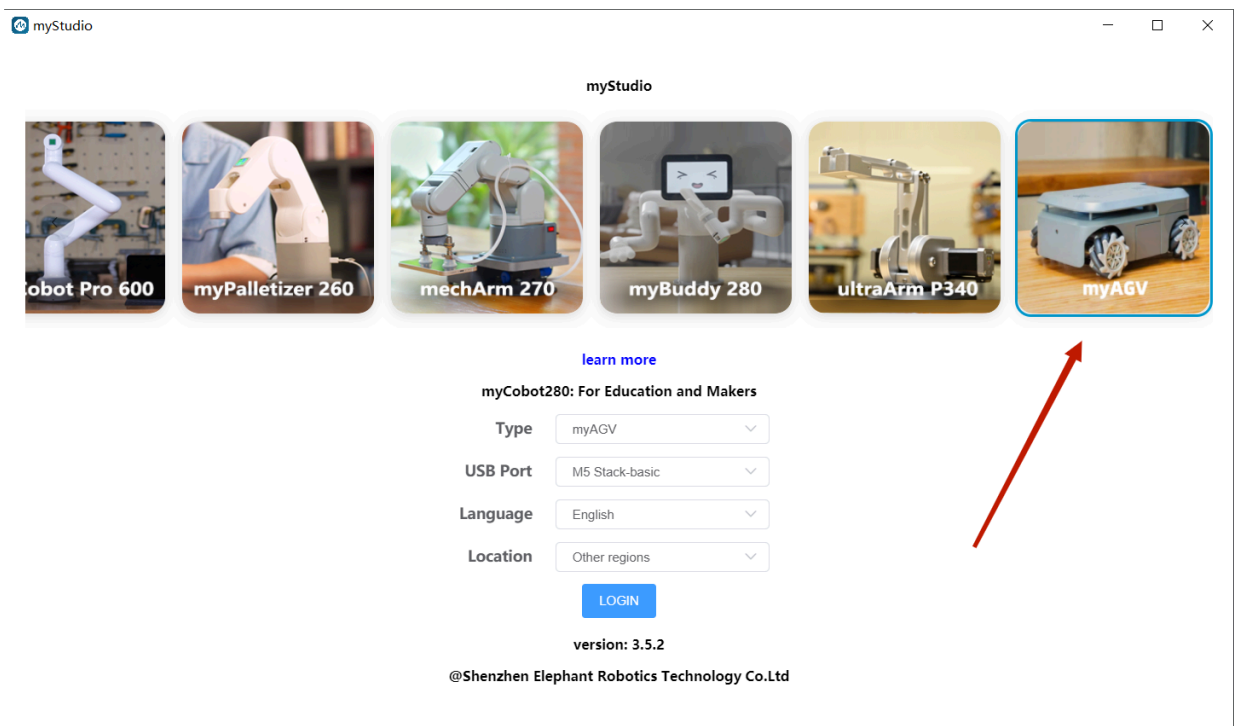
2.4 myAGV Firmware burning

1. Burning Pico Firmware: picoMain

Step 1: Turn on the computer and connect the screen.

For detailed tutorials, please refer to: [2.5.1 First-time Use · GitBook \(elephantrobotics.com\)](#)

Step 2: Select the myAGV image, select the model language and region, and click to log in.



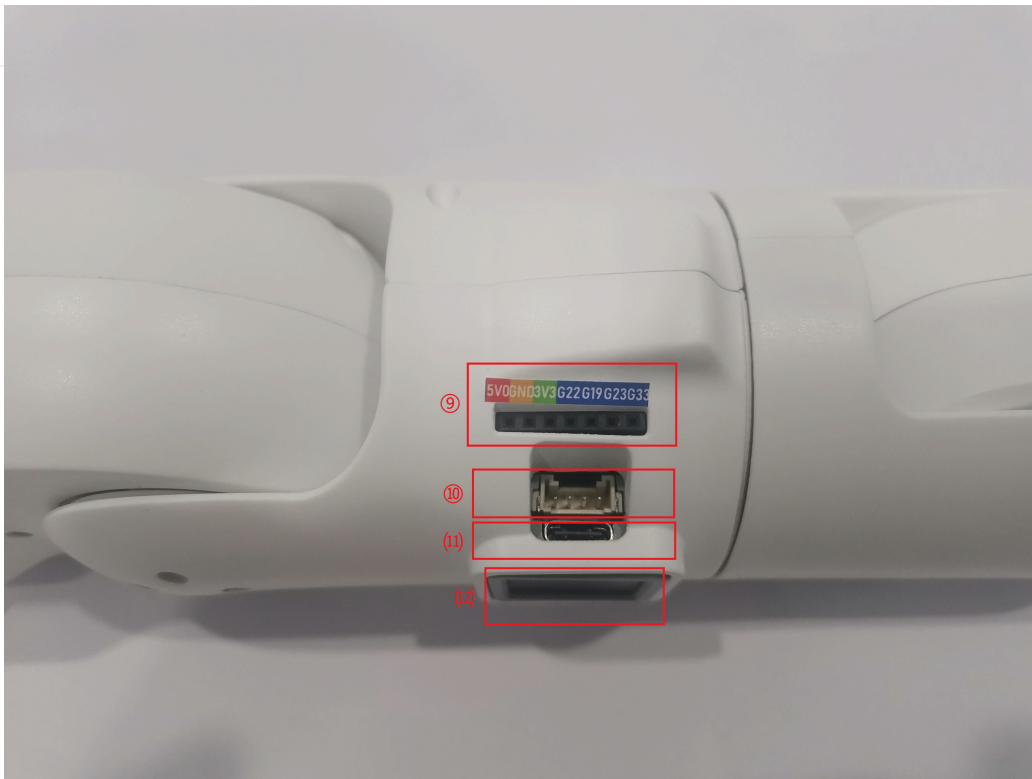
Step 3: After logging in, click 'Basic' and select picoMain to burn.



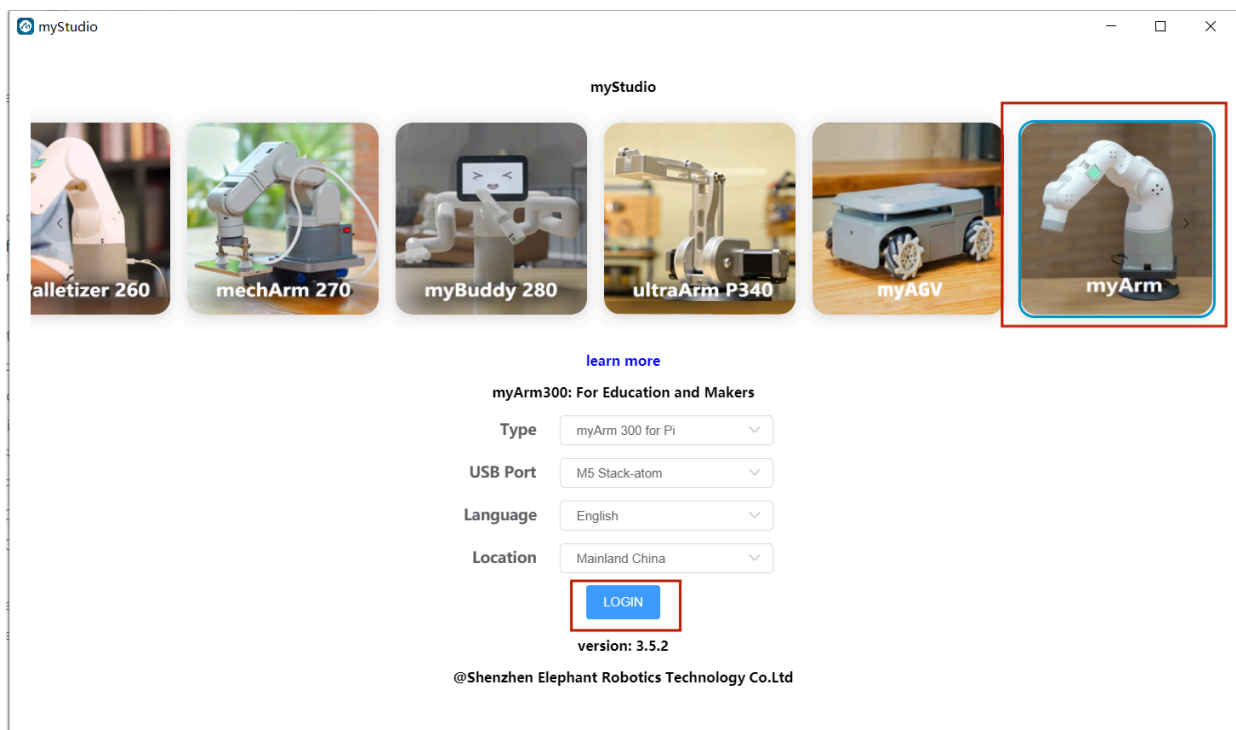
2.5 myArm Firmware burning

1. Burning Atom Firmware

Step 1: Connects to a PC. Use USB to connect the Atom interface (labeled 11 as Atom Type C interface).



Step 2: Select the myArm image, select the model language and region, and click to login.



Step 3: After login, click 'Basic', select AtomMain, click download, and then click the burn button to burn.

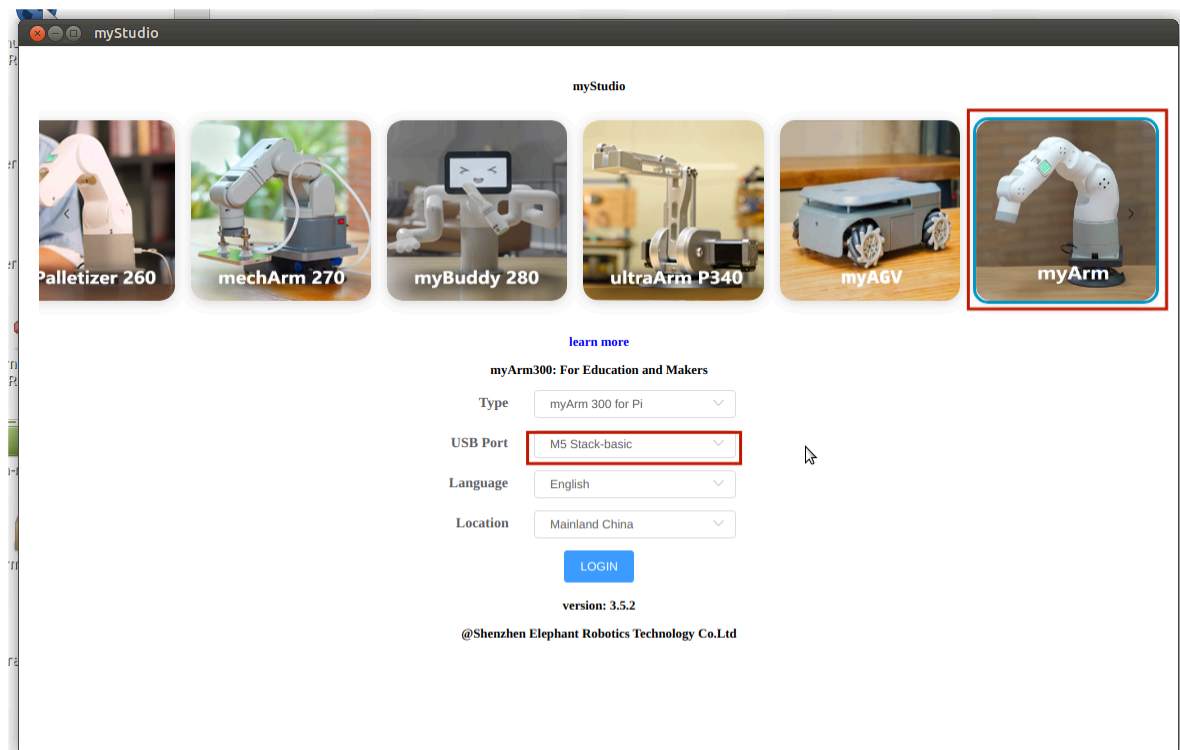


2. Burning Pico Firmware: picoMain

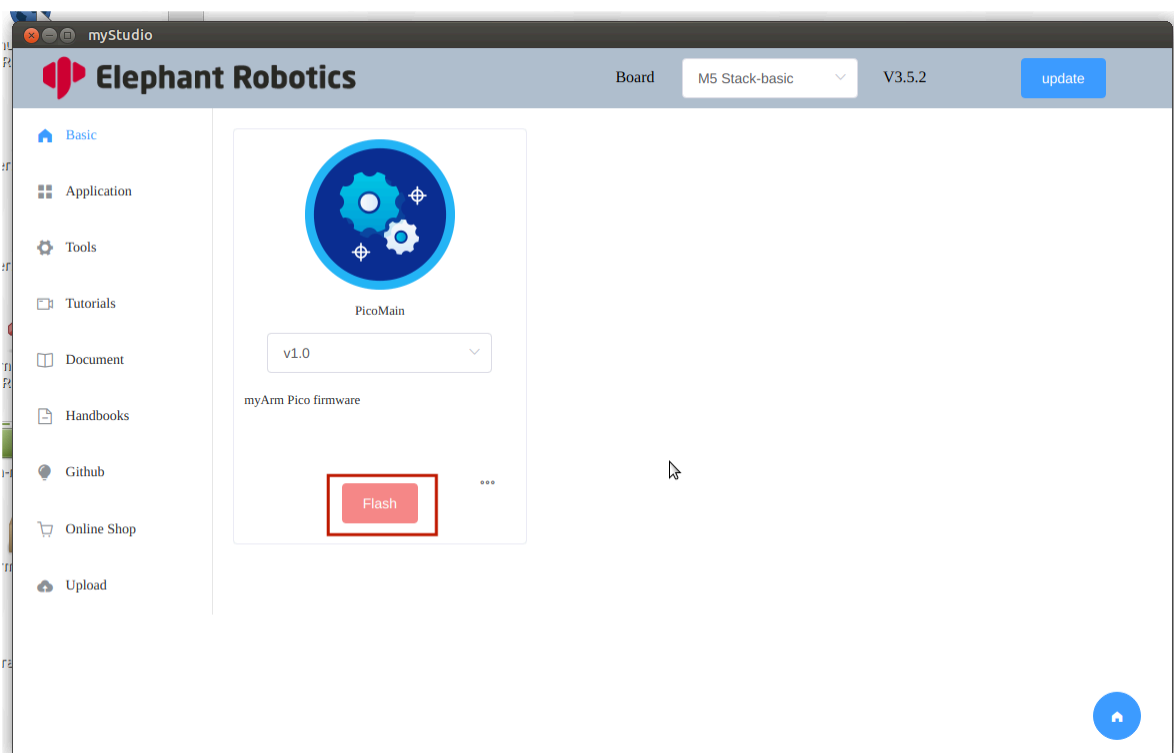
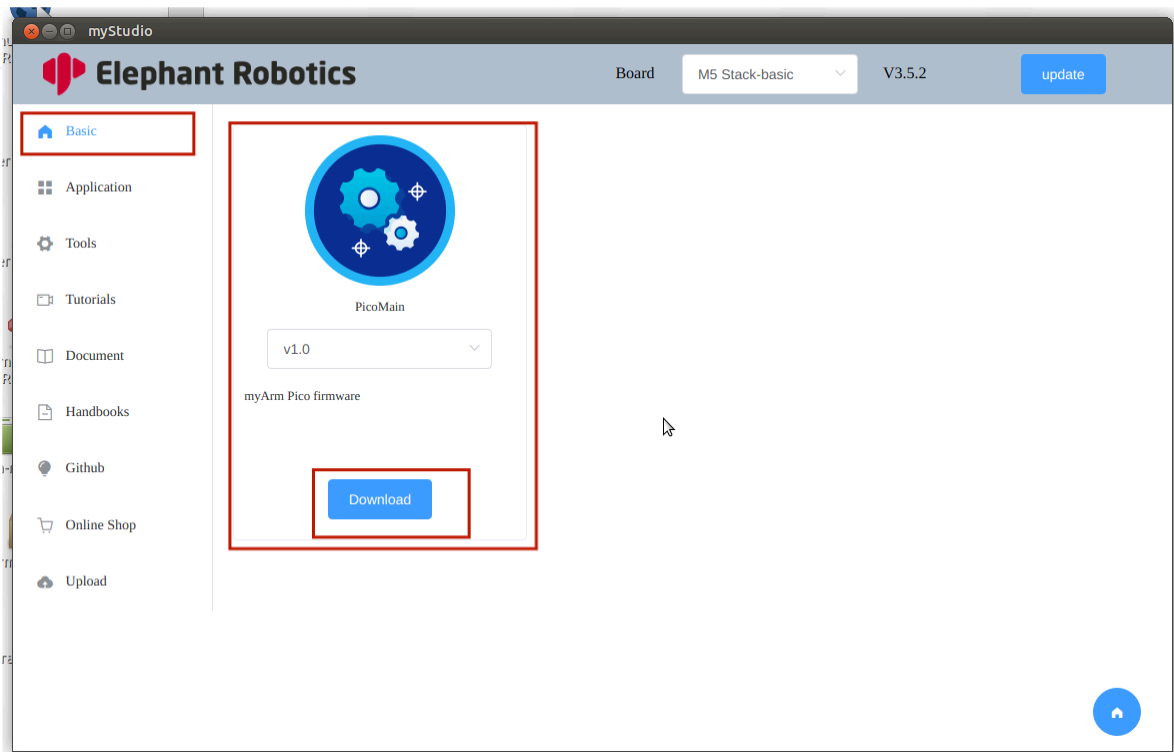
Step 1: Turn on the computer and connect the screen.

For detailed tutorials, please refer to: [initial-setup](#)

Step 2: Select the myArm image, select the model language and region, and click to login.



Step 3: After login, click 'Basic', select picoMain, click download, and then click the burn button to burn.



Factory firmware introduction

We divide the devices into two categories: **micro-controller** and **micro-CPU**. (Only introduce myCobot series, myPalletizer series, mechArm series products)

- **Micro-controller devices**

- **myCobot 280-M5**
- **myCobot 320-M5**
- **myPalletizer 260- M5**
- **mechArm 270-M5**

- **Micro-CPU devices**

- **myCobot 280-Pi**
- **myCobot 320-Pi**
- **mechArm 270-Pi**

The difference between the micro-CPU and the micro-controller mainly focuses on three aspects: hardware structure, application fields and instruction set characteristics:

- **Hardware structure.** The micro-CPU is a single-chip CPU, while the micro-controller integrates a CPU and other circuits in an integrated circuit chip to form a complete microcomputer system. Beside the CPU, the micro-controller includes a RAM, ROM, serial interface, parallel interface, timer, and interrupt scheduling circuit.
- **Application fields.** The micro-CPU is usually used as a CPU in a microcomputer system, and they are designed for such application. This is just the advantage of the micro-CPU. However, the micro-controller is usually used in control-oriented applications, where system design seeks to miniaturize and minimize the number of components. The micro-controller is suitable for those applications in which control of input/output devices is implemented with very few components, while the micro-CPU is suitable for information processing in a computer system.
- **Instruction set characteristics.** Due to different applications, the instruction set of the micro-controller is also different from the one of the micro-CPU. The instruction set of the micro-CPU enhances the processing function, making it have a powerful addressing mode and the instructions suitable for manipulating large amounts of data. The instructions of the micro-CPU allow operation of nibbles, bytes, words, and even double words. By using address pointers and offsets, the micro-CPU provides an addressing mode that allows access of large amounts of data. The auto-increment and auto-decrement modes make it easy to access data in the units of bytes, words, or double words.

Here you can gradually familiarize yourself with the robot arm through simple routines.

1 Factory firmware for micro-controller devices - miniRoboFlow

miniRoboFlow has four main functions:

- **Maincontrol**

- **Robot drag teaching:** The operator can drag robot joints directly to make them do ideal postures, and then save the actions in the robot through **button operation**. The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.

- **Calibration**

- Calibrating the robot arm is the precondition for precise control of the robot arm, and setting joint zero and initializing the potential of the motor are basic jobs for subsequent advanced development.
- **Transponder**
 - Communication timeliness is vital to the micro-controller robot arm. For such arm, we often send control instructions to **M5Stack-basic** at the bottom. Through communication forwarding, the end effector analyzes the instructions and then implements target actions. At present, **myCobot280** has three methods of communication: **serial port**, **Bluetooth**, and **WIFI**.
- **Information**
 - Link test is a detection function that uses the motor in the robot arm and the connection state of **Atom**. The function allows the user to remove equipment faults easily. During the link test, the connection state of the equipment for the robot arm, including the **connection of the servo** and the **communication state of Atom** can be seen. In micro-controller devices, the versions of their current firmwares are shown on M5Stack-basic.

2 Factory firmware for micro-CPU devices- python demo

At present, the python demos that have been opened to micro-CPU devices are:

- **drag_trial_teaching**
 - **drag teaching** The operator can drag robot joints directly to make them do ideal postures, and then save the actions in the robot through **button operation**. The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.
- **rasp_mycobot_test_gui**
 - This python demo is a test tool for micro-CPU devices, including functions **robot arm calibration** and **connection detection**.
 - **Robot arm calibration** Calibrating the robot arm is the precondition for precise control of the robot arm, and setting joint zero and initializing the potential of the motor are basic jobs for subsequent advanced development.
 - **Connection detection** Link test is a detection function that uses the motor in the robot arm and the connection state of **Atom**. The function allows the user to remove equipment faults easily. During the link test, the connection state of the equipment for the robot arm, including the **connection of the servo** and the **communication state of Atom** can be seen. In micro-controller devices, the versions of their current firmwares are shown on M5Stack-basic.

Drag teaching



Robot drag teaching is a process during which the operator can drag the joints of the robot directly to make them do ideal postures, and then make records corresponding thereto.

The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.

Different types of equipment have different operating methods. They have the approximate steps below:

- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic**. Choose the **Maincontrol** function (It is unnecessary to burn **M5Stack-basic** for micro-CPU devices).
- Press the recording button/keyboard key
- select a storage path (micro-CPU devices haven't this step)
- directly drag the joints of the robot arm to move them to your desired positions to complete a set of movements
- press the designated button/keyboard key for saving
- press the play button/keyboard key/keyboard key
- select a corresponding storage path, and then the robot arm starts to move
- and finally press the exit button/keyboard key to exit this function

In this chapter, we will teach you how to get started easily and experience the fun of cobot drag teaching.

Drag-and-Record Demonstration

1 Applicable Robotic Arms

- myCobot 280 Pi
- myCobot 320 Pi
- mechArm 270 Pi
- myArm 300 Pi
- myPalletizer 260

. 2 Steps to Operate the Robotic Arms

Step 1: Burn the latest version of **atomMain** for **Atom**.

Step 2: Create a Python file on the desktop named as `drag_trial_teaching.py` and copy the following codes.

Go to [GitHub](#) to copy the codes.

```

import time
import os
import sys
import termios
import tty
import threading
import json
import serial
import serial.tools.list_ports

from pymycobot.mycobot280 import MyCobot280
from pymycobot.mycobot320 import MyCobot320
from pymycobot.mecharm270 import MechArm270
from pymycobot.myarm import MyArm
from pymycobot.mypalletizer260 import MyPalletizer260

port: str
mc: MyCobot280
sp: int = 80

def setup():
    global port, mc

    print("")

    print("1 - MyCobot280")
    print("2 - MyCobot320")
    print("3 - MechArm270")
    print("4 - MyArm300")
    print("5 - MyPalletizer260")
    _in = input("Please input 1 - 5 to choose:")
    robot_model = None
    if _in == "1":
        robot_model = MyCobot280
        print("MyCobot280\n")
        print("Please enter the model type:")
        print("1. Pi")
        print("2. Jetson Nano")
        print("Default is Pi")
        model_type = input()

        if model_type == "2":
            port = "/dev/ttyTHS1"
        else:
            pass
    elif _in == "2":
        robot_model = MyCobot320
        print("MyCobot320\n")

```

```

elif _in == "3":
    robot_model = MechArm270
    print("MechArm270\n")

elif _in == "4":
    robot_model = MyArm
    print("MyArm300\n")

elif _in == '5':
    robot_model = MyPalletizer260
    print('MyPalletizer260\n')

else:
    print("Please choose from 1 - 5.")
    print("Exiting...")
    exit()

plist = list(serial.tools.list_ports.comports())
idx = 1
for port in plist:
    print("{} : {}".format(idx, port))
    idx += 1

if not plist:
    pass
else:
    _in = input("\nPlease input 1 - {} to choice:".format(idx - 1))
    port = str(plist[int(_in) - 1]).split(" - ")[0].strip()
    print(port)
    print("")

    baud = 1000000
    _baud = input("Please input baud(default:1000000):")
    try:
        baud = int(_baud)
    except Exception:
        pass
    print(baud)
    print("")

    DEBUG = False
    f = input("Wether DEBUG mode[Y/n](default:n):")
    if f in ["y", "Y", "yes", "Yes"]:
        DEBUG = True

    mc = robot_model(port, baud, debug=DEBUG)
    mc.power_on()

class Raw(object):
    """Set raw input mode for device"""

```

```

def __init__(self, stream):
    self.stream = stream
    self.fd = self.stream.fileno()

def __enter__(self):
    self.original_stty = termios.tcgetattr(self.stream)
    tty.setcbreak(self.stream)

def __exit__(self, type, value, traceback):
    termios.tcsetattr(self.stream, termios.TCSANOW, self.original_stty)

class Helper(object):
    def __init__(self) -> None:
        self.w, self.h = os.get_terminal_size()

    def echo(self, msg):
        print("\r{}".format(" " * self.w), end="")
        print("\r{}".format(msg), end="")

class TeachingTest(Helper):
    def __init__(self, mycobot) -> None:
        super().__init__()
        self.mc = mycobot
        self.recording = False
        self.playing = False
        self.record_list = []
        self.record_t = None
        self.play_t = None
        self.path = os.path.dirname(os.path.abspath(__file__)) + "/record.txt"

    def record(self):
        self.record_list = []
        self.recording = True

        # self.mc.set_fresh_mode(0)
        if isinstance(self.mc, MyCobot320):
            self.mc.release_all_servos(1)
        else:
            self.mc.release_all_servos()

    def _record():
        while self.recording:
            start_t = time.time()
            if isinstance(self.mc, (MyArm, MyPalletizer260)):
                angles = self.mc.get_encoders()
                interval_time = time.time() - start_t
                if angles:

```

```

        record = [angles, interval_time]
        self.record_list.append(record)
        print("\r {}".format(time.time() - start_t), end="")
    else:
        angles = self.mc.get_encoders()
        speeds = self.mc.get_servo_speeds()
        interval_time = time.time() - start_t
        if angles and speeds:
            record = [angles, speeds, interval_time]
            self.record_list.append(record)
            print("\r {}".format(time.time() - start_t), end="")

    self.echo("Start recording.")
    self.record_t = threading.Thread(target=_record, daemon=True)
    self.record_t.start()

def stop_record(self):

    if isinstance(self.mc, MyArm):
        self.mc.power_on()
    else:
        self.mc.focus_all_servos()

    if self.recording:
        self.recording = False
        self.record_t.join()
        self.echo("Stop record")

def play(self):
    self.echo("Start play")
    if isinstance(self.mc, (MyArm, MyPalletizer260)):
        for record in self.record_list:
            encoders, interval_time = record
            self.mc.set_encoders(encoders, 100)
            time.sleep(interval_time)
    else:
        i = 0
        for record in self.record_list:
            encoders, speeds, interval_time = record
            self.mc.set_encoders_drag(encoders, speeds)
            if i == 0:
                time.sleep(3)
            i += 1
            time.sleep(interval_time)
        self.echo("Finish play")

def loop_play(self):
    self.playing = True

    def _loop():

```



```

        while self.playing:
            self.play()

        self.echo("Start loop play.")
        self.play_t = threading.Thread(target=_loop, daemon=True)
        self.play_t.start()

    def stop_loop_play(self):
        if self.playing:
            self.playing = False
            self.play_t.join()
            self.echo("Stop loop play.")

    def save_to_local(self):
        if not self.record_list:
            self.echo("No data should save.")
            return
        with open(self.path, "w") as f:
            json.dump(self.record_list, f, indent=2)
            self.echo("save dir: {}".format(self.path))

    def load_from_local(self):
        with open(self.path, "r") as f:
            try:
                data = json.load(f)
                self.record_list = data
                self.echo("Load data success.")
            except Exception:
                self.echo("Error: invalid data.")

    def print_menu(self):
        print(
            """\
\r q: quit
\r r: start record
\r c: stop record
\r p: play once
\r P: loop play / stop loop play
\r s: save to local
\r l: load from local
\r f: release mycobot
\r-----
            """
        )

    def start(self):
        self.print_menu()

        while not False:
            with Raw(sys.stdin):

```

```

key = sys.stdin.read(1)

if key == "q":
    break
elif key == "r": # recorder
    self.record()
elif key == "c": # stop recorder
    self.stop_record()
elif key == "p": # play
    if not self.playing:
        self.play()
    else:
        print("Already playing. Please wait till current play stop or stop loop play.")
elif key == "P": # loop play
    if not self.playing:
        self.loop_play()
    else:
        self.stop_loop_play()
elif key == "s": # save to local
    self.save_to_local()
elif key == "l": # load from local
    self.load_from_local()
elif key == "f": # free move
    if isinstance(self.mc, MyCobot320):
        self.mc.release_all_servos(1)
    else:
        self.mc.release_all_servos()
    self.echo("Released")
else:
    print(key)
    continue

if __name__ == "__main__":
    setup()
    recorder = TeachingTest(mc)
    recorder.start()

```

Step 3: Open the terminal(CTRL+ALT+T) and type the following command:

```

cd Desktop
python3 drag_trial_teaching.py

```

```
er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:
```

Step 4: Select the model number `Please input 1 - 5 to choose:` , enter 4 here and press Enter.

```
er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:4
MyArm300

1 : /dev/ttyAMA0 - ttyAMA0

Please input 1 - 1 to choice:
```

Step 5: Type 1 and press `Enter` .

```

er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:4
MyArm300

1 : /dev/ttyAMA0 - ttyAMA0

Please input 1 - 1 to choice:1
/dev/ttyAMA0

Please input baud(default:1000000):

```

Step 6: Type the baud rate and press `Enter`.

- *myCobot 280-Pi: 1000000*
- *myCobot 320-Pi: 115200*
- *myCobot 270-Pi: 1000000*
- *myArm 300-Pi: 115200*
- *myPalletizer 260-Pi: 1000000*

```

er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:4
MyArm300

1 : /dev/ttyAMA0 - ttyAMA0

Please input 1 - 1 to choice:1
/dev/ttyAMA0

Please input baud(default:1000000):115200
115200

Wether DEBUG mode[Y/n](default:n):

```

Step 7: The question that Whether DEBUG mode appears on the screen. Type Y or N.

```
er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:4
MyArm300

1 : /dev/ttyAMA0 - ttyAMA0

Please input 1 - 1 to choice:1
/dev/ttyAMA0

Please input baud(default:1000000):115200
115200

Wether DEBUG mode[Y/n](default:n):n
q: quit
r: start record
c: stop record
p: play once
P: loop play / stop loop play
s: save to local
l: load from local
f: release mycobot
-----
```

Step 8: Type **r** to start recording, and then you can move the joints.

Notice: Type **q** to exit this program.

```
er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:4
MyArm300

1 : /dev/ttyAMA0 - ttyAMA0

Please input 1 - 1 to choice:1
/dev/ttyAMA0

Please input baud(default:1000000):115200
115200

Wether DEBUG mode[Y/n](default:n):n
q: quit
r: start record
c: stop record
p: play once
P: loop play / stop loop play
s: save to local
l: load from local
f: release mycobot
-----
0.017798185348510742
```

Step 9: Type **c** to stop recording.

```
er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:4
MyArm300

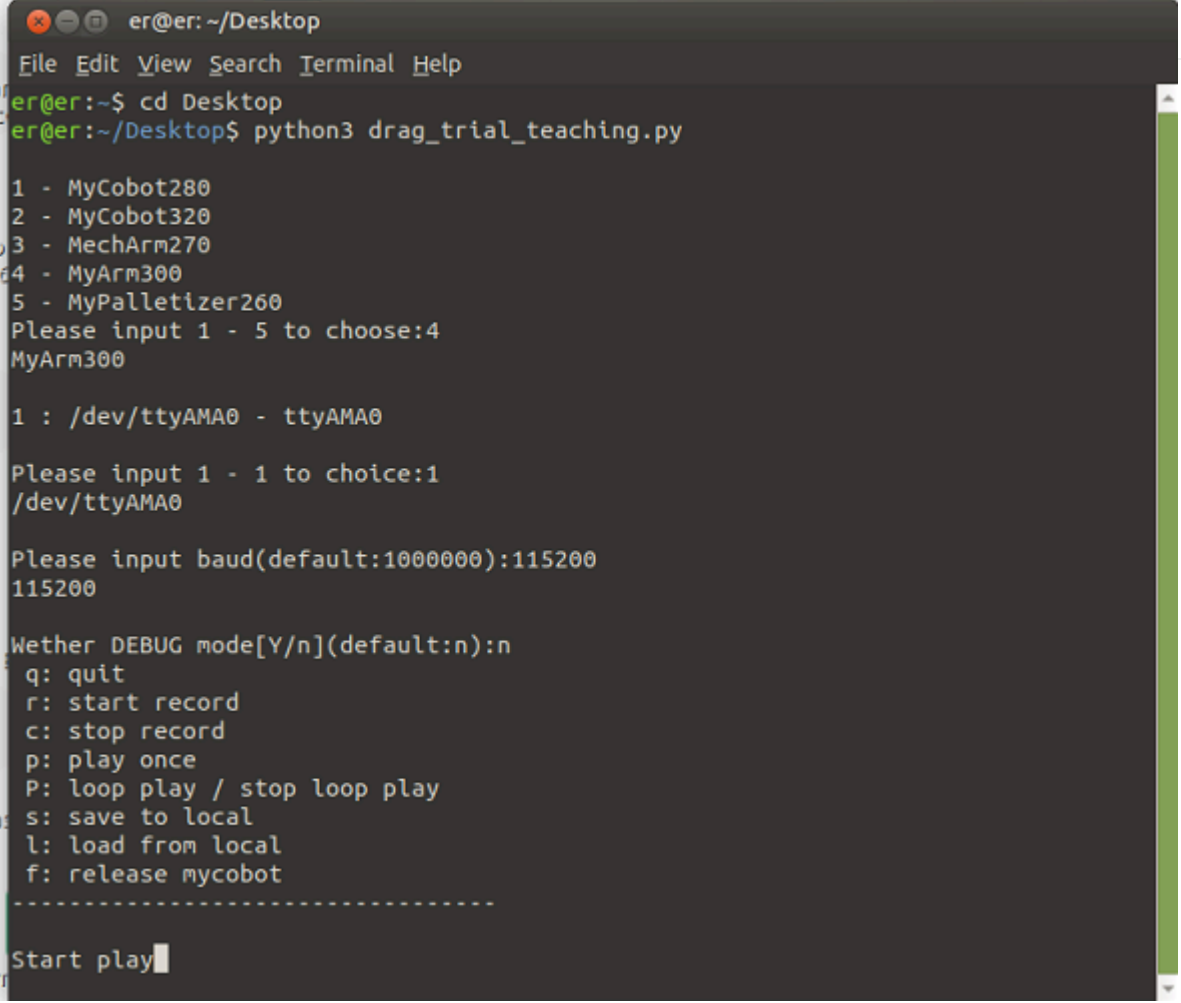
1 : /dev/ttyAMA0 - ttyAMA0

Please input 1 - 1 to choice:1
/dev/ttyAMA0

Please input baud(default:1000000):115200
115200

Wether DEBUG mode[Y/n](default:n):n
q: quit
r: start record
c: stop record
p: play once
P: loop play / stop loop play
s: save to local
l: load from local
f: release mycobot
-----
Stop record
```

Step 10: Type **p** (lowercase letters) to play once.

A terminal window titled 'er@er: ~/Desktop' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of a Python script 'drag_trial_teaching.py'. The script lists five robot models: MyCobot280, MyCobot320, MechArm270, MyArm300, and MyPalletizer260. It prompts the user to choose one (1-5), and '4' is entered, selecting 'MyArm300'. Next, it prompts for a device path, and '/dev/ttyAMA0' is entered. Then, it prompts for a baud rate (default: 1000000), and '115200' is entered. A menu of options is displayed: 'q: quit', 'r: start record', 'c: stop record', 'p: play once', 'P: loop play / stop loop play', 's: save to local', 'l: load from local', and 'f: release mycobot'. A dashed line separates the menu from the 'Start play' prompt, which has a cursor at the end.

```
er@er: ~/Desktop
File Edit View Search Terminal Help
er@er:~$ cd Desktop
er@er:~/Desktop$ python3 drag_trial_teaching.py

1 - MyCobot280
2 - MyCobot320
3 - MechArm270
4 - MyArm300
5 - MyPalletizer260
Please input 1 - 5 to choose:4
MyArm300

1 : /dev/ttyAMA0 - ttyAMA0

Please input 1 - 1 to choice:1
/dev/ttyAMA0

Please input baud(default:1000000):115200
115200

Wether DEBUG mode[Y/n](default:n):n
q: quit
r: start record
c: stop record
p: play once
P: loop play / stop loop play
s: save to local
l: load from local
f: release mycobot
-----
Start play
```

Type **p** (uppercase letters) to play in a loop / stop loop play.

Type **f** to release each joint of the robot arm (This function can be used to stop the movement of the robot arm, and it is used when all joints are locked).

Robot arm calibration

Note: This operation has been performed before the default robot is delivered. You do not need to repeat the operation. Incorrect use of this function may cause damage to the robot. If your robot works without abnormal conditions, **please do not use it**. Thank you for your cooperation.



Calibrating the robot arm is the precondition for precise control of the robot arm, and setting joint zero and initializing the potential of the motor are basic jobs for subsequent advanced development.

Different types of equipment have different operating methods. They have the approximate steps below:

- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic**.
- Choose the **Calibration** function (It is unnecessary to burn **Basic** for micro-CPU devices)
- move all joints of the robot arm to their zero positions (align them with the scale line of zero positions)
- Press the calibration button and get started with robot arm calibration
- press the test button to test the zero positions
- press the exit button to exit this function

In this chapter, we will teach you how to calibrate the robot arm and to test and verify the joints.

Calibration

1 Applicable Robotic Arms

- myCobot 280 Pi
- myCobot 320 Pi
- mechArm 270 Pi

2 Steps to Operate the Arms

Step 1: Burn the latest **atomMain** for **Atom**.

Step 2: Create a Python file on the desktop named as `rasp_mycobot_test_gui.py` and copy the following codes.

Go to [GitHub](#) to copy the codes.

```

#!/usr/bin/env python3

import socket
import tkinter
from tkinter import ttk
import time
import threading
import os
import textwrap
import serial
import serial.tools.list_ports

from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD

LOG_NUM = 0

class MycobotTest(object):
    def __init__(self):
        self.mycobot = None

        self.win = tkinter.Tk()
        self.win.title("树莓派版 Mycobot 测试工具")
        self.win.geometry("918x511+10+10") # 290 160为窗口大小, +10 +10 定义窗口弹出时的默认展示位置

        self.port_label = tkinter.Label(self.win, text="选择串口: ")
        self.port_label.grid(row=0)
        self.port_list = ttk.Combobox(
            self.win, width=15, postcommand=self.get_serial_port_list
        ) # 创建下拉菜单
        self.get_serial_port_list() # 给下拉菜单设定值
        self.port_list.current(0)
        self.port_list.grid(row=0, column=1)

        self.baud_label = tkinter.Label(self.win, text="选择波特率: ")
        self.baud_label.grid(row=1)
        self.baud_list = ttk.Combobox(self.win, width=15)
        self.baud_list["value"] = ("115200", "1000000")
        self.baud_list.current(0)
        self.baud_list.grid(row=1, column=1)

        # Connect
        self.connect_label = tkinter.Label(self.win, text="连接mycobot: ")
        self.connect_label.grid(row=2)
        self.connect = tkinter.Button(self.win, text="连接", command=self.connect_mycobot)
        self.disconnect = tkinter.Button(
            self.win, text="断开", command=self.disconnect_mycobot
        )
        self.connect.grid(row=3)

```

```

self.disconnect.grid(row=3, column=1)

# Check servo.
self.check_label = tkinter.Label(self.win, text="检测连接: ")
self.check_label.grid(row=4)
self.check_btn = tkinter.Button(
    self.win, text="开始检测", command=self.check_mycobot_servos
)
self.check_btn.grid(row=4, column=1)

# Calibration.
self.calibration_num = None
self.calibration_label = tkinter.Label(self.win, text="校准舵机: ")
self.calibration_label.grid(row=5)
self.calibration_btn = tkinter.Button(
    self.win, text="开始校准", command=self.calibration_mycobot
)
self.calibration_btn.grid(row=5, column=1)

# LED.
self.set_color_label = tkinter.Label(self.win, text="测试Atom灯板: ")
self.set_color_label.grid(row=6, columnspan=2)
self.color_red = tkinter.Button(
    self.win, text="设置红色", command=lambda: self.send_color("red")
)
self.color_green = tkinter.Button(
    self.win, text="设置绿色", command=lambda: self.send_color("green")
)
self.color_red.grid(row=7)
self.color_green.grid(row=7, column=1)

# Aging test.
self.aging_stop = False
self.movement_label = tkinter.Label(self.win, text="老化循环动作: ")
self.movement_label.grid(row=8)
self.start_btn = tkinter.Button(
    self.win, text="开始", command=self.start_aging_test
)
self.start_btn.grid(row=9)
self.stop_btn = tkinter.Button(
    self.win, text="停止", command=self.stop_aging_test
)
self.stop_btn.grid(row=9, column=1)

# Release
self.release_btn = tkinter.Button(
    self.win, text="放松所有电机", command=self.release_mycobot
)
self.release_btn.grid(row=10)

```

```

# rectify
self.rectify_btn = tkinter.Button(
    self.win, text="校准pid", command=self.rectify_mycobot
)
self.rectify_btn.grid(row=10, column=1)

# Log output.
self.log_label = tkinter.Label(self.win, text="日志: ")
self.log_label.grid(row=0, column=12)
_f = tkinter.Frame(self.win)
_bar = tkinter.Scrollbar(_f, orient=tkinter.VERTICAL)
self.log_data_Text = tkinter.Text(
    _f, width=100, height=35, yscrollcommand=_bar.set
)
_bar.pack(side=tkinter.RIGHT, fill=tkinter.Y)
_bar.config(command=self.log_data_Text.yview)
self.log_data_Text.pack()
# self.log_data_Text.grid(row=1, column=12, rowspan=15, columnspan=10)
_f.grid(row=1, column=12, rowspan=15, columnspan=10)

def run(self):
    self.win.mainloop() # run

# =====
# Connect method
# =====
def connect_mycobot(self):
    self.port = port = self.port_list.get()
    if not port:
        self.write_log_to_Text("请选择串口")
        return
    self.baud = baud = self.baud_list.get()
    if not baud:
        self.write_log_to_Text("请选择波特率")
        return
    baud = int(baud)

    try:
        # self.mycobot = MyCobot(PI_PORT, PI_BAUD)
        self.mycobot = MyCobot(port, baud)
        # self.mycobot = MyCobot("/dev/cu.usbserial-0213245D", 115200)
        self.write_log_to_Text("连接成功 !")
    except Exception as e:
        err_log = """\
        \r连接失败 !!!
        \r=====
        {}
        \r=====
        """.format(
            e

```

```

    )
    self.write_log_to_Text(err_log)

def disconnect_mycobot(self):
    if not self.has_mycobot():
        return

    try:
        del self.mycobot
        self.mycobot = None
        self.write_log_to_Text("断开连接成功 !")
    except AttributeError:
        self.write_log_to_Text("还没有连接mycobot! ! ! ")

# =====
# Function method
# =====

def release_mycobot(self):
    if not self.has_mycobot():
        return

    self.mycobot.release_all_servos()
    self.write_log_to_Text("Release over.")

def check_mycobot_servos(self):
    if not self.has_mycobot():
        return

    ping_commands = [
        [255, 255, 1, 2, 1, 251],
        [255, 255, 2, 2, 1, 250],
        [255, 255, 3, 2, 1, 249],
        [255, 255, 4, 2, 1, 248],
        [255, 255, 5, 2, 1, 247],
        [255, 255, 6, 2, 1, 246],
        [255, 255, 7, 2, 1, 246],
    ]

    res = []
    for idx, command in enumerate(ping_commands, start=1):
        self.mycobot._write(command)
        time.sleep(0.1)
        if not self.mycobot._read(1):
            res.append(idx)
        time.sleep(0.1)

    if res:
        self.write_log_to_Text("关节 {} 无法通信! ! ! ".format(res))
    else:
        self.write_log_to_Text("所有关节连接正常。")

def calibration_mycobot(self):

```

```

"""Calibration button click event.

Click to calibrate one motor at a time and calibrate in turn. After all
calibration, resume initialization.
"""

if not self.has_mycobot():
    return

if not self.calibration_num:
    self.calibration_num = 0

self.calibration_num += 1

self.mycobot.set_servo_calibration(self.calibration_num)
time.sleep(0.1)
self.mycobot.send_angle(self.calibration_num, 0, 0)
time.sleep(0.1)
self.write_log_to_Text("校准电机 %s 结束." % self.calibration_num)

if self.calibration_num == 6:
    self.write_log_to_Text("全部校准完成.")
    self.calibration_num = None
    self.rectify_mycobot()
    self._calibration_test()

def send_color(self, color: str):
    if not self.has_mycobot():
        return

    color_dict = {
        "red": [255, 0, 0],
        "green": [0, 255, 0],
        "blue": [0, 0, 255],
    }
    self.mycobot.set_color(*color_dict[color])
    self.write_log_to_Text("发送颜色: {}".format(color))

def start_aging_test(self):
    if not self.has_mycobot():
        return

    self.aging_stop = False
    self.aging = threading.Thread(target=self._aging_test, daemon=True)
    self.aging.start()
    # self._aging_test()
    self.write_log_to_Text("开始循环老化测试 ...")

def stop_aging_test(self):
    try:
        os.system("sudo systemctl stop aging_test.service")
        os.system("sudo rm /home/pi/aging_test.sh")

```



```

        os.system("sudo rm /home/pi/Desktop/aging_test.py")
        os.system("sudo rm /etc/systemd/system/aging_test.service")
        os.system("sudo systemctl daemon-reload")
        self.write_log_to_Text("结束循环老化测试.")
    except:
        self.write_log_to_Text("结束老化测试失败 !!! ")

def rectify_mycobot(self):
    if not self.has_mycobot():
        return

    for i in range(1, 7):
        self.mycobot.set_servo_data(i, 24, 0)
        time.sleep(0.1)
        self.mycobot.set_servo_data(i, 26, 3)
        time.sleep(0.1)
        self.mycobot.set_servo_data(i, 27, 3)
        time.sleep(0.1)
    time.sleep(0.1)
    for i in range(1, 7):
        self.write_log_to_Text(
            "Read servo {} pid data, 24:{}, 26:{}, 27:{}`".format(
                i,
                self.mycobot.get_servo_data(i, 24),
                self.mycobot.get_servo_data(i, 26),
                self.mycobot.get_servo_data(i, 27),
            )
        )
        time.sleep(0.1)

# =====
# Utils method
# =====

def has_mycobot(self):
    """Check whether it is connected on mycobot"""
    if not self.mycobot:
        self.write_log_to_Text("还没有连接mycobot!!! ")
        return False
    return True

def _aging_test(self):
    """
    Aging test thread target.
    By using in `start_aging_test()` and `stop_aging_test()`.
    """
    # if socket.gethostname() != "pi":
    #     self.write_log_to_Text("老化测试支持 Raspberry OS.")
    #     return

    aging_test_content_py = textwrap.dedent(

```

```

"""\

#!/usr/bin/python3

from pymycobot.mycobot import MyCobot

from pymycobot import PI_PORT, PI_BAUD

import time

mycobot = MyCobot('%s', %s)

def aging_test():

    # fast

    mycobot.set_color(255, 0, 0)

    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([170, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([-170, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 90, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, -90, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 140, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, -140, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 130, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, -110, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 165, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, -165, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, 0], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, 180], 95)
    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, -180], 95)

    # middle

    mycobot.set_color(0, 255, 0)

    mycobot.wait(3).send_angles([0, 0, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([170, 0, 0, 0, 0, 0], 55)
    mycobot.wait(6.5).send_angles([-170, 0, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 90, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, -90, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 140, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, -140, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 130, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, -110, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 165, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, -165, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 0], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 180], 55)
    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, -180], 55)

    # slow

    mycobot.set_color(0, 0, 255)

    mycobot.wait(5).send_angles([0, 0, 0, 0, 0, 0], 15)

```

```

        mycobot.wait(7).send_angles([170, 0, 0, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([-170, 0, 0, 0, 0, 0], 15)
        mycobot.wait(11).send_angles([0, 0, 0, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([0, 90, 0, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([0, -90, 0, 0, 0, 0], 15)
        mycobot.wait(0).send_angles([0, 0, 0, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([0, 0, 140, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([0, 0, -140, 0, 0, 0], 15)
        mycobot.wait(11).send_angles([0, 0, 0, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([0, 0, 0, 130, 0, 0], 15)
        mycobot.wait(7).send_angles([0, 0, 0, -110, 0, 0], 15)
        mycobot.wait(11).send_angles([0, 0, 0, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([0, 0, 0, 0, 165, 0], 15)
        mycobot.wait(7).send_angles([0, 0, 0, 0, -165, 0], 15)
        mycobot.wait(11).send_angles([0, 0, 0, 0, 0, 0], 15)
        mycobot.wait(7).send_angles([0, 0, 0, 0, 0, 180], 15)
        mycobot.wait(7).send_angles([0, 0, 0, 0, 0, -180], 15)

    if __name__ == '__main__':
        while True:
            aging_test()

    """
    % (self.prot, self.baud)
)

aging_test_content_sh = textwrap.dedent(
    """\
    #!/bin/bash

    /usr/bin/python3 /home/pi/Desktop/aging_test.py
    """
)

aging_test_content_service = textwrap.dedent(
    """\
    [Unit]
    Description=aging-test

    [Service]
    Type=forking
    User=pi
    Restart=on-failure
    RestartSec=2
    ExecStart=/home/pi/aging_test.sh

    [Install]
    WantedBy=multi-user.target
    """
)

os.system(
    'echo "' + aging_test_content_py + '" >> /home/pi/Desktop/aging_test.py'
)

os.system('echo "' + aging_test_content_sh + '" >> /home/pi/aging_test.sh')

```

```

os.system("sudo chmod +x /home/pi/aging_test.sh")
os.system(
    'echo "'
    + aging_test_content_service
    + '" >> /home/pi/Desktop/aging_test.service'
)
os.system(
    "sudo mv /home/pi/Desktop/aging_test.service /etc/systemd/system/aging_test.service"
)
os.system("sudo systemctl enable aging_test.service")
os.system("sudo systemctl start aging_test.service")

def _calibration_test(self):
    self.write_log_to_Text("开始测试校准.")
    angles = [0, 0, 0, 0, 0, 0]
    test_angle = [-20, 20, 0]
    for i in range(6):
        for j in range(3):
            angles[i] = test_angle[j]
            self.mycobot.send_angles(angles, 0)
            time.sleep(0.7)
    self.write_log_to_Text("测试校准结束.")

def get_serial_port_list(self):
    plist = [
        str(x).split(" - ")[0].strip() for x in serial.tools.list_ports.comports()
    ]
    print(plist)
    self.port_list["value"] = plist
    return plist

def get_current_time(self):
    """Get current time with format."""
    current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(time.time()))
    return current_time

def write_log_to_Text(self, logmsg: str):
    global LOG_NUM
    current_time = self.get_current_time()
    logmsg_in = str(current_time) + " " + str(logmsg) + "\n" # 换行

    if LOG_NUM <= 18:
        self.log_data_Text.insert(tkinter.END, logmsg_in)
        LOG_NUM += len(logmsg_in.split("\n"))
        # print(LOG_NUM)
    else:
        self.log_data_Text.insert(tkinter.END, logmsg_in)
        self.log_data_Text.yview("end")

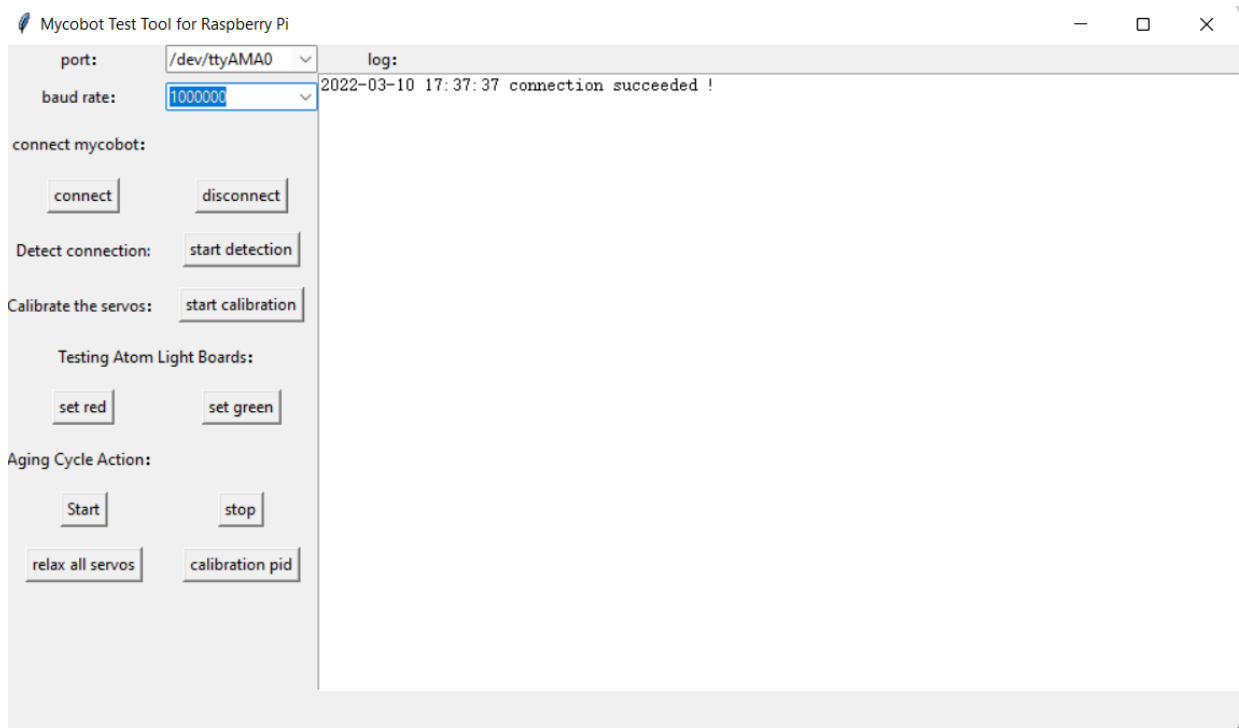
```

```
if __name__ == "__main__":  
    MycobotTest().run()
```

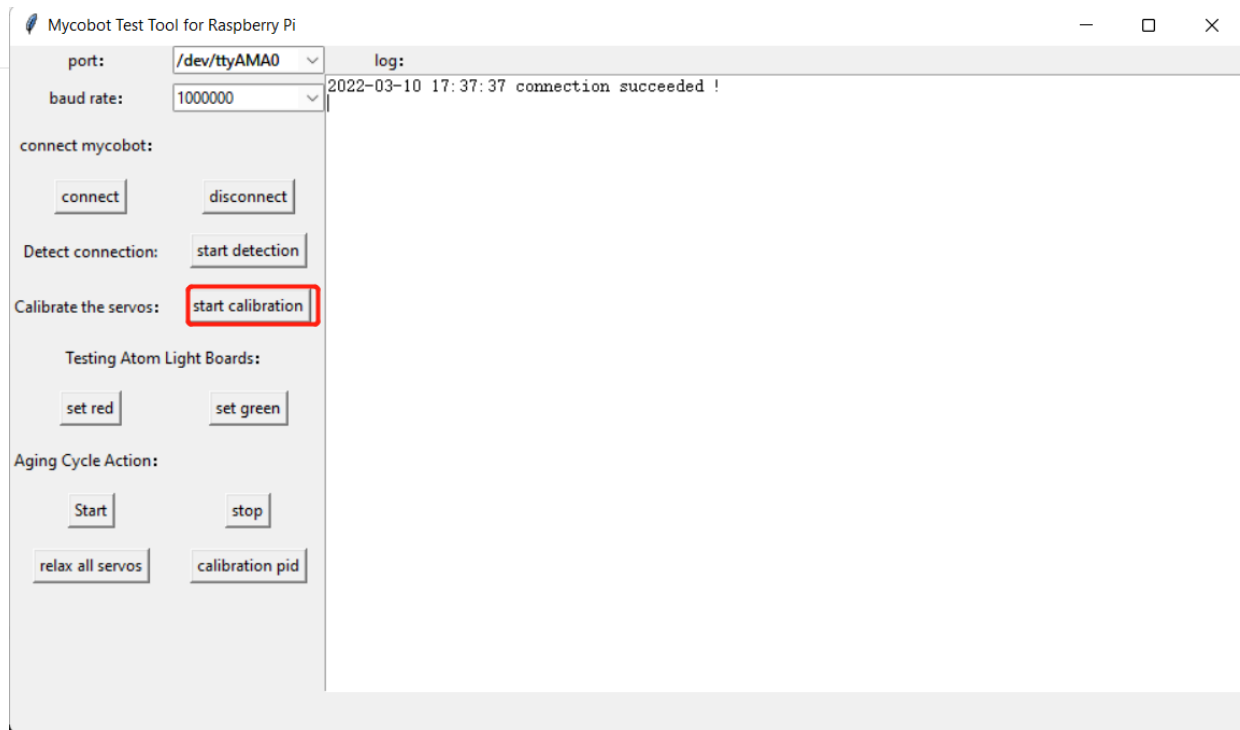
Step 3: Open the terminal (CTRL+ALT+T), and then type the following commands:

```
cd Desktop  
python3 drag_trial_teaching.py
```

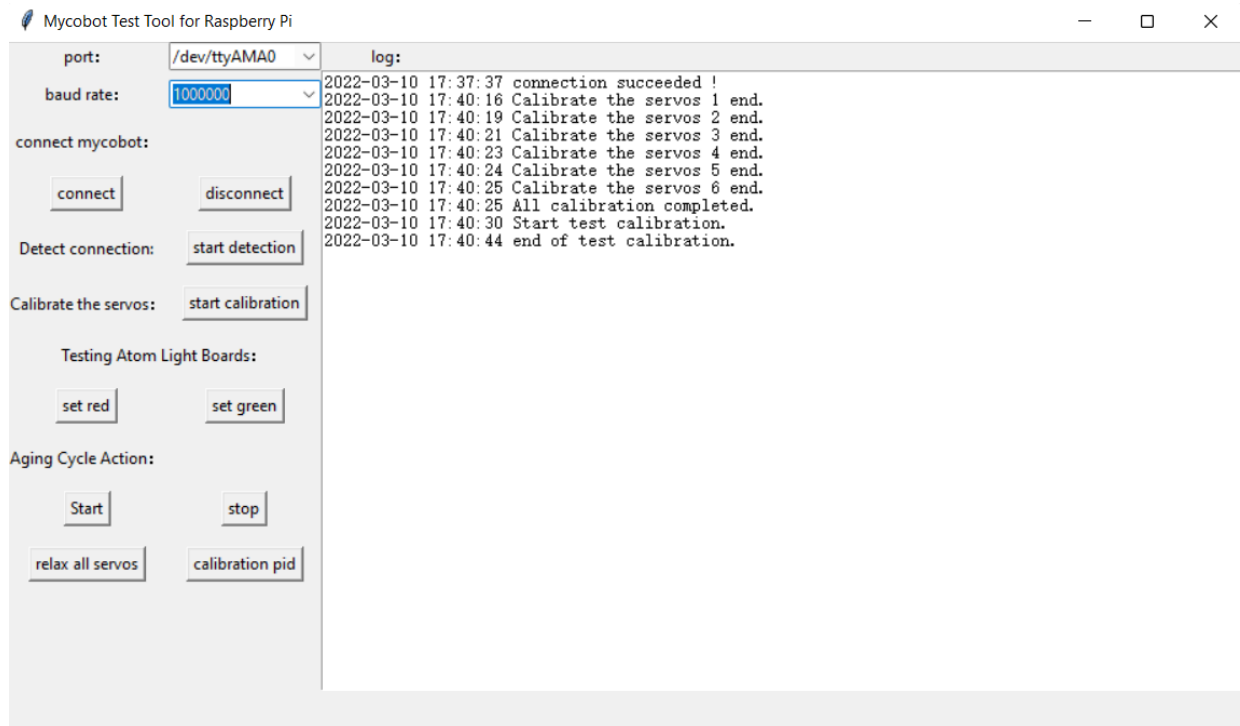
Step 4: Select baud rate and then click on `connect` .



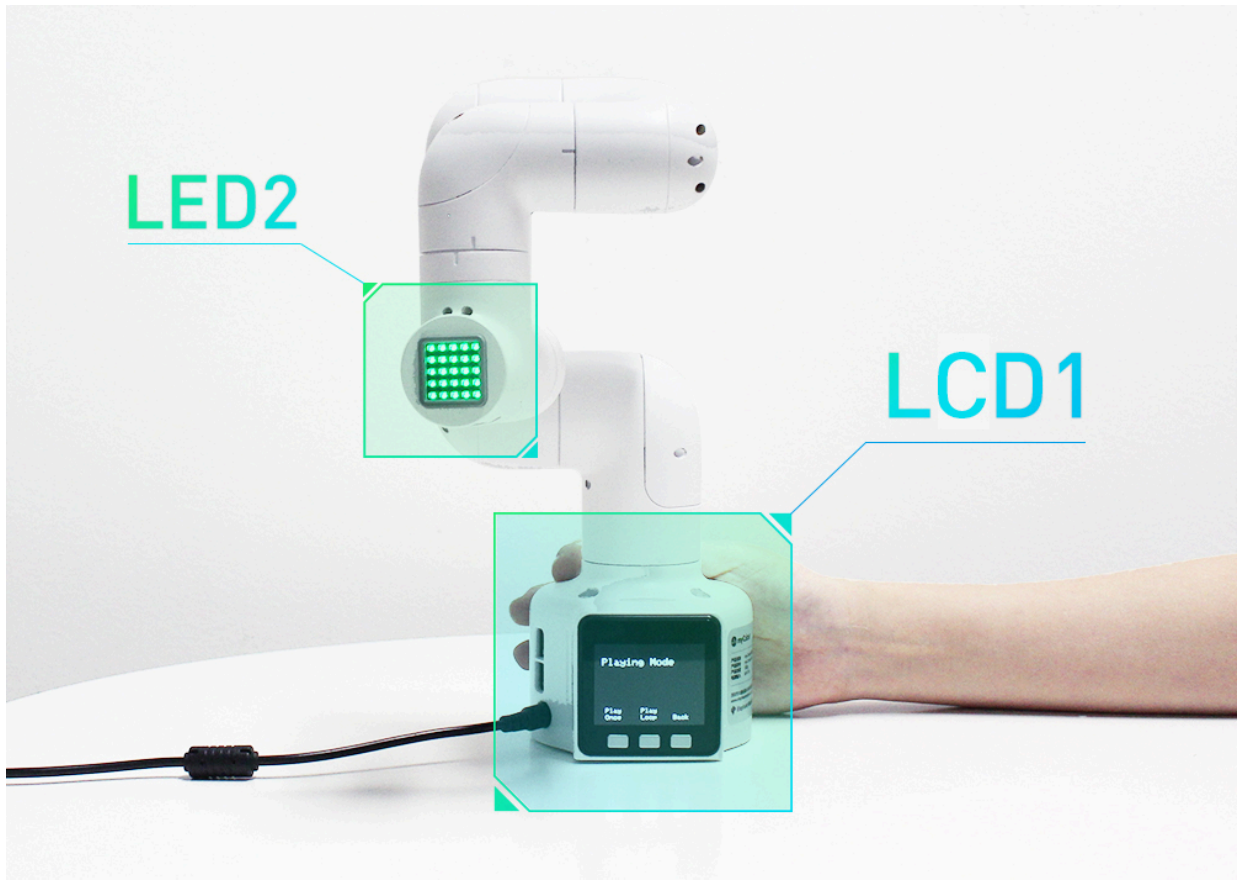
Step 5: Set all joints to the zero positions, and click on **Start calibration**.



Step 6: When **Calibration completed** is shown on the screen, the calibration is finished.



Communication forwarding



Communication timeliness is vital to the micro-controller robot arm. For such arm, we often send control instructions to M5Stack-basic of the stand. Through communication forwarding, the end effector analyzes the instructions and then implements target actions.

This function is mainly used for the customer to develop robot arms independently in different environments.

Different types of equipment have different operating methods. They have the approximate steps below:

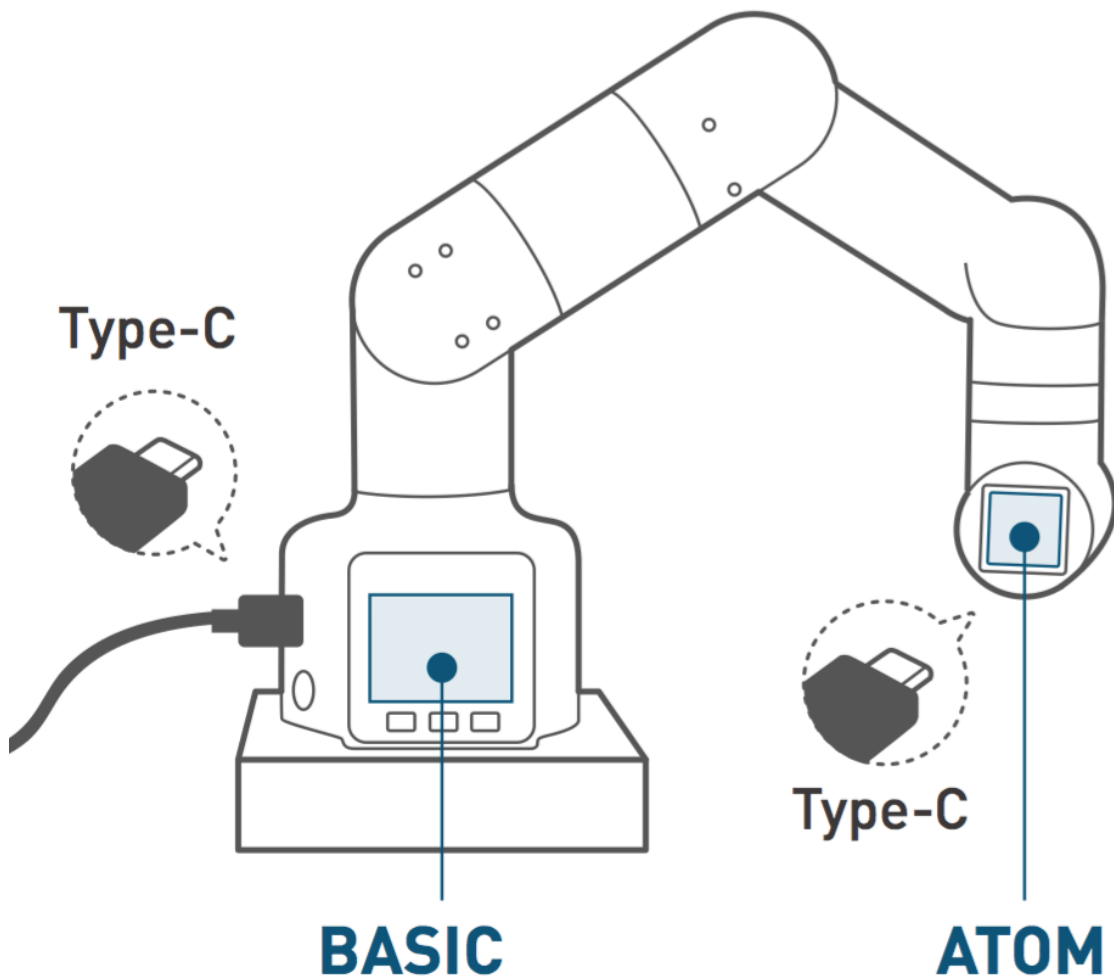
- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic**.
- Choose the **Transponder** function (It is unnecessary to burn **M5Stack-basic** for micro-CPU devices)
- press the detection key to detect whether the communication between M5Stack-basic and the end effector Atom is normal
- press the exit button to exit this function.

In this chapter, we can perform real-time detection of the communication between M5Stack-basic and the end effector Atom to see whether it is normal.

Remote Control

- Currently available [VNC](#) and [SSH](#)

Connection detection



Link test is a detection function that uses the motor in the robot arm and the connection state of **Atom**. The function allows the user to remove equipment faults easily.

During the link test, the connection state of the equipment for the robot arm, including the **connection of the servo** and the **communication state of Atom** can be seen. In **micro-controller devices**, the versions of their current firmwares are shown on M5Stack-basic.

Different types of devices have different operating methods. They have the approximate steps below:

- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic**.
- Choose the **Information** function (It is unnecessary to burn **M5Stack-basic** for micro-CPU devices)
- press the detection key to detect the connection of the devices
- press the firmware view key to check the version of the current firmware
- press the exit button to exit this function.

In this chapter, we will learn how to use the detection function for different types of devices.

Connection Detection

1 Applicable Robotic Arm

- myCobot 280 Pi
- myCobot 320 Pi
- mechArm 270 Pi

2 Steps to Operate

Step 1: Burn the latest **atomMain** for **Atom**.

Step 2: Create a new Python file named as `rasp_mycobot_test_gui.py` and copy the following codes.

Go to [GitHub](#) to download the codes.

```

#!/usr/bin/env python3

import tkinter
from tkinter import ttk
import time
import serial
import serial.tools.list_ports

from pymycobot.mycobot import MyCobot

LOG_NUM = 0

class MycobotTest(object):
    def __init__(self):
        self.mycobot = None

        self.win = tkinter.Tk()
        self.win.title("RaspberryPI myCobot Test Tool")
        self.win.geometry("918x480+10+10")

        self.port_label = tkinter.Label(self.win, text="Port: ")
        self.port_label.grid(row=0)
        self.port_list = ttk.Combobox(
            self.win, width=15, postcommand=self.get_serial_port_list
        )
        self.get_serial_port_list()
        self.port_list.current(0)
        self.port_list.grid(row=0, column=1)

        self.baud_label = tkinter.Label(self.win, text="Baudrate: ")
        self.baud_label.grid(row=1)
        self.baud_list = ttk.Combobox(self.win, width=15)
        self.baud_list["value"] = ("1000000", "115200")
        self.baud_list.current(1)
        self.baud_list.grid(row=1, column=1)

        # Connect
        self.connect_label = tkinter.Label(self.win, text="Connect myCobot: ")
        self.connect_label.grid(row=2)
        self.connect = tkinter.Button(self.win, text="connect", command=self.connect_mycobot)
        self.disconnect = tkinter.Button(
            self.win, text="disconnect", command=self.disconnect_mycobot
        )
        self.connect.grid(row=3)
        self.disconnect.grid(row=3, column=1)

        # Check servo.
        self.check_label = tkinter.Label(self.win, text="Check connection: ")
        self.check_label.grid(row=4)
        self.check_btn = tkinter.Button(
            self.win, text="Start to check", command=self.check_mycobot_servos

```

```

    )

    self.check_btn.grid(row=4, column=1)

    # LED.
    self.set_color_label = tkinter.Label(self.win, text="Test Atom LED: ")
    self.set_color_label.grid(row=5, columnspan=2)
    self.color_red = tkinter.Button(
        self.win, text="Set RED", command=lambda: self.send_color("red")
    )
    self.color_green = tkinter.Button(
        self.win, text="Set GREEN", command=lambda: self.send_color("green")
    )
    self.color_red.grid(row=6)
    self.color_green.grid(row=6, column=1)

    # Log output.
    self.log_label = tkinter.Label(self.win, text="Log: ")
    self.log_label.grid(row=0, column=12)
    _f = tkinter.Frame(self.win)
    _bar = tkinter.Scrollbar(_f, orient=tkinter.VERTICAL)
    self.log_data_Text = tkinter.Text(
        _f, width=100, height=35, yscrollcommand=_bar.set
    )
    _bar.pack(side=tkinter.RIGHT, fill=tkinter.Y)
    _bar.config(command=self.log_data_Text.yview)
    self.log_data_Text.pack()
    # self.log_data_Text.grid(row=1, column=12, rowspan=15, columnspan=10)
    _f.grid(row=1, column=12, rowspan=15, columnspan=10)

def run(self):
    self.win.mainloop() # run

# =====
# Connect method
# =====

def connect_mycobot(self):
    self.prot = port = self.port_list.get()
    if not port:
        self.write_log_to_Text("Please choose port")
        return
    self.baud = baud = self.baud_list.get()
    if not baud:
        self.write_log_to_Text("Please choose baudrate")
        return
    baud = int(baud)

    try:
        # self.mycobot = MyCobot(PI_PORT, PI_BAUD)
        self.mycobot = MyCobot(port, baud)
        time.sleep(0.5)

```

```

        self.mycobot._write([255,255,3,22,1,250])
        time.sleep(0.5)
        # self.mycobot = MyCobot("/dev/cu.usbserial-0213245D", 115200)
        self.write_log_to_Text("Connect successfully !")
    except Exception as e:
        err_log = """\
            \r连接失败 !!!
            \r=====
            {}
            \r=====
        """.format(
            e
        )
        self.write_log_to_Text(err_log)

def disconnect_mycobot(self):
    if not self.has_mycobot():
        return

    try:
        del self.mycobot
        self.mycobot = None
        self.write_log_to_Text("Disconnect !")
    except AttributeError:
        self.write_log_to_Text("Please connect myCobot first! ! ! ")

# =====
#  Function method
# =====

def check_mycobot_servos(self):
    if not self.has_mycobot():
        return

    res = []
    for i in range(1,8):
        _data = self.mycobot.get_servo_data(i , 5)
        time.sleep(0.02)
        if _data != i:
            res.append(i)

    if res:
        self.write_log_to_Text("Joint {} lose connection! ! ! ".format(res))
    else:
        self.write_log_to_Text("All joints' connections are normal。")

def send_color(self, color: str):
    if not self.has_mycobot():
        return

    color_dict = {

```

```

        "red": [255, 0, 0],
        "green": [0, 255, 0],
        "blue": [0, 0, 255],
    }

    self.mycobot.set_color(*color_dict[color])
    self.write_log_to_Text("Send color: {}".format(color))

# =====
# Utils method
# =====

def has_mycobot(self):
    """Check whether it is connected on mycobot"""
    if not self.mycobot:
        self.write_log_to_Text("Please connect myCobot first! ! ! ")
        return False
    return True

def get_serial_port_list(self):
    plist = [
        str(x).split(" - ")[0].strip() for x in serial.tools.list_ports.comports()
    ]
    print(plist)
    self.port_list["value"] = plist
    return plist

def get_current_time(self):
    """Get current time with format."""
    current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(time.time()))
    return current_time

def write_log_to_Text(self, logmsg: str):
    global LOG_NUM
    current_time = self.get_current_time()
    logmsg_in = str(current_time) + " " + str(logmsg) + "\n"

    if LOG_NUM <= 18:
        self.log_data_Text.insert(tkinter.END, logmsg_in)
        LOG_NUM += len(logmsg_in.split("\n"))
        # print(LOG_NUM)
    else:
        self.log_data_Text.insert(tkinter.END, logmsg_in)
        self.log_data_Text.yview("end")

if __name__ == "__main__":
    MycobotTest().run()

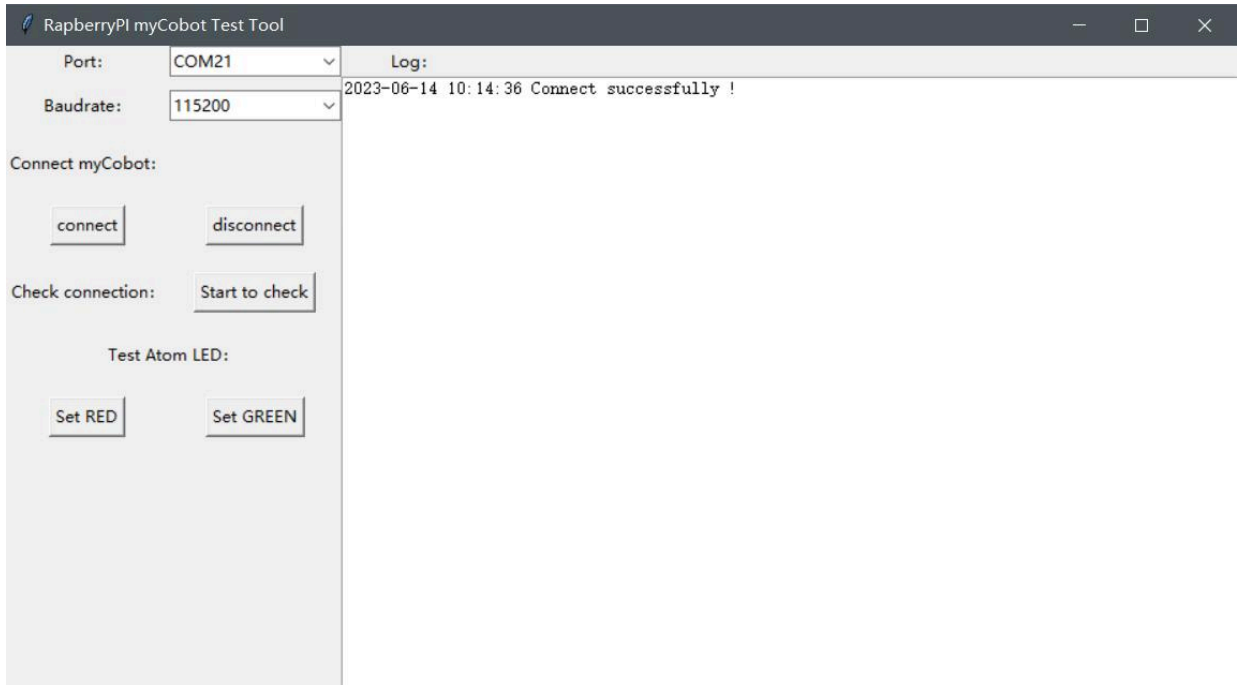
```

Step 3: Open the terminal (CTRL+ALT+T) and type the following commands. And then, click on ENTER:

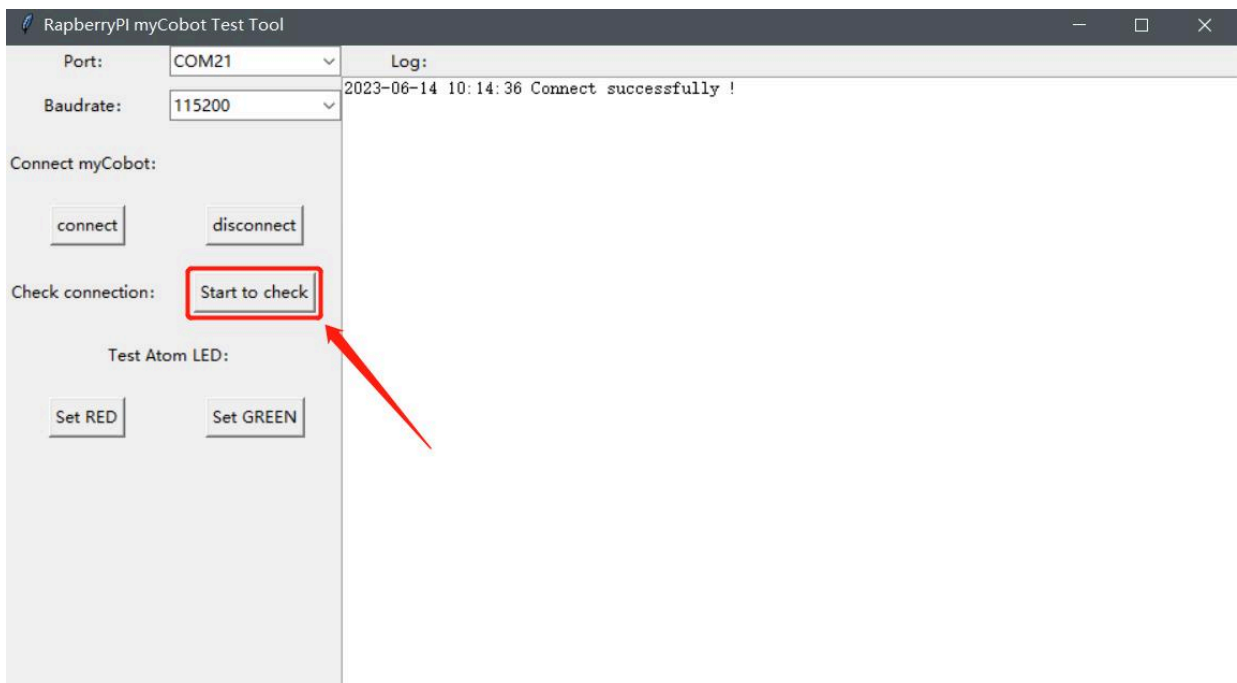
```
cd Desktop
python3 rasp_mycobot_test_gui.py
```

Step 4: Choose the corresponding serial port and baud rate at the left top corner. After that, choose the baud rate corresponding to the device, and click on **Connect**.

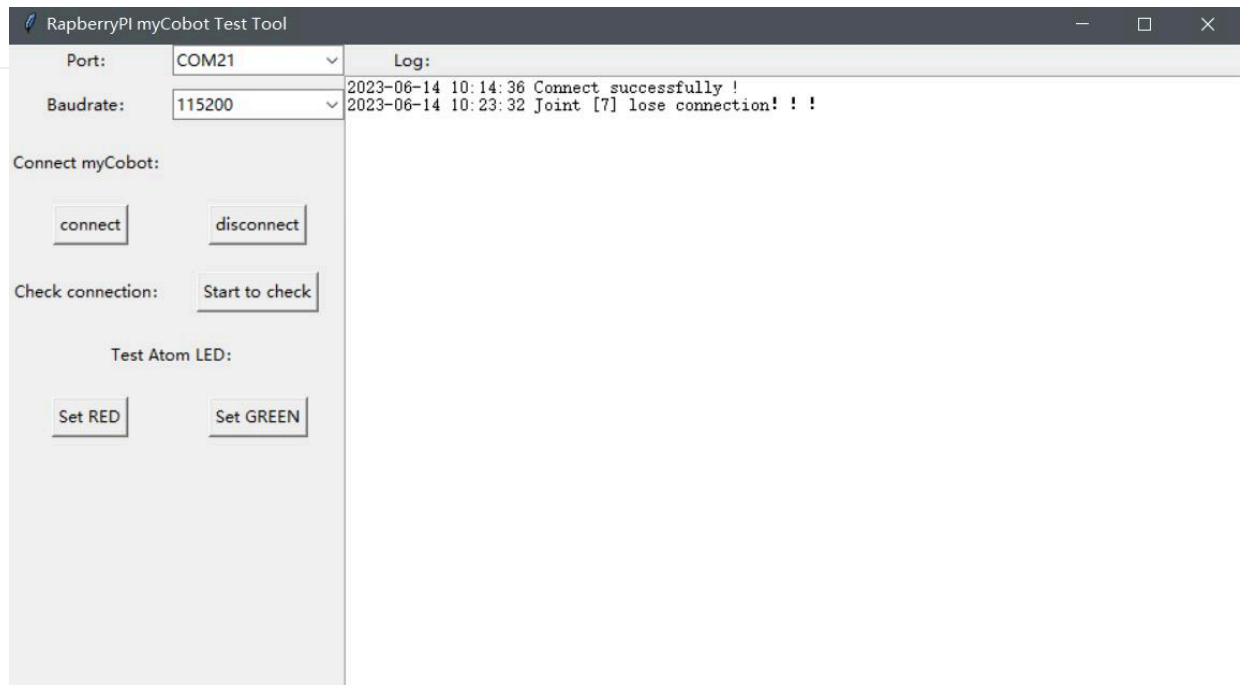
- *myCobot 280 Pi:1000000*
- *myCobot 320 Pi:115200*



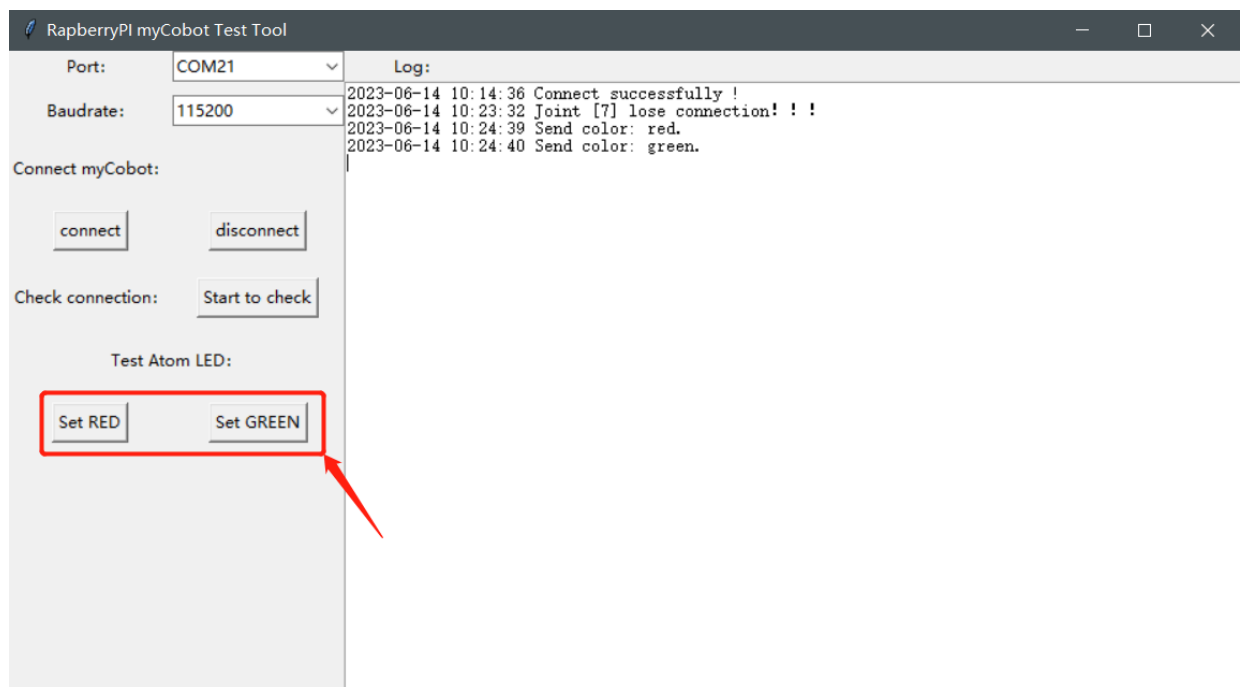
Step 5: Set all joints of the robot arm to the zero position, and click on **Start to check**

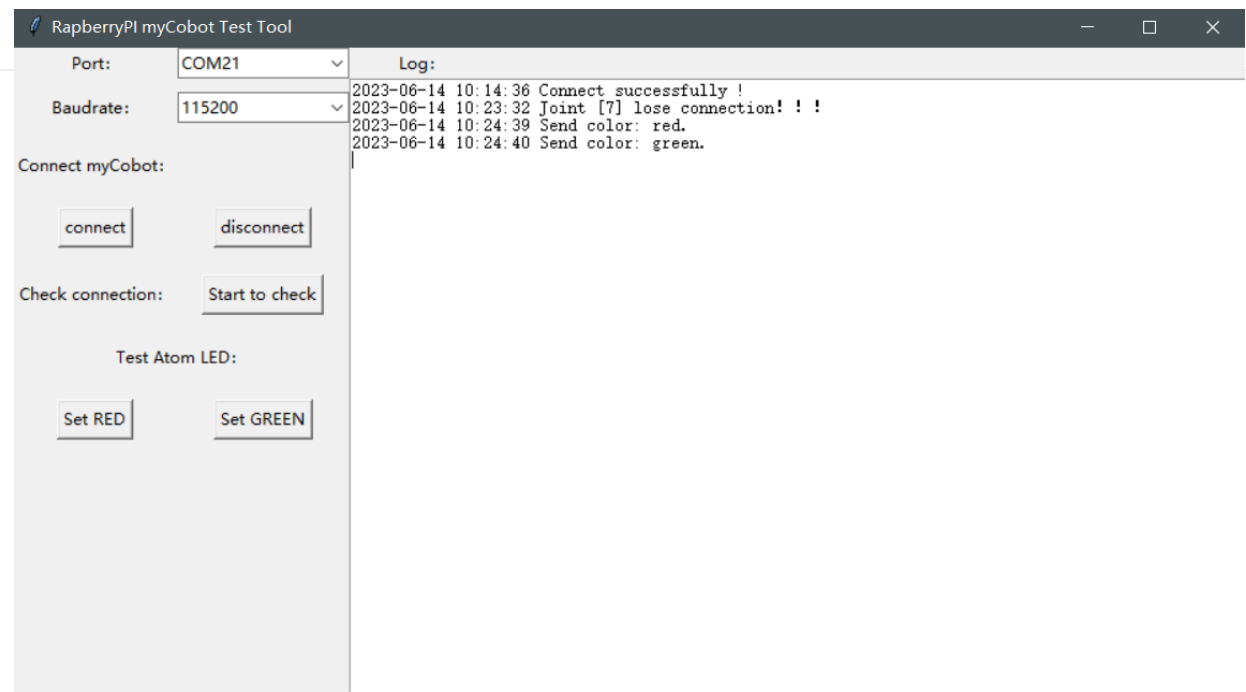


Step 6: When the prompt `Joint [7] lose connection!!!`, which means that joint 7 fails to communicate, the detection completes.



Step 7: These two buttons are able to change the color of Atom LED.





Use for the first time

Step 1: Unpacking

After receiving the packing box of the robot, first check that the box is intact. **If it is lost or damaged, contact the logistics company and the local supplier.** After opening the box, check the inside items by referring to the packing list.



1 Items included in myCobot robot arm [standard kit]

- myCobot robot arm (Model: myCobot -280)
- Product catalog
- Power supply
- USB-Type C
- Jumper
- M4*35, cup head hexagon socket, full thread, stainless steel screw
- Allen wrench

2 Confirming the operational environment and indicators



Install the robot system in the environment that meets the conditions described in the table in order to give full play to and maintain the performance of this machine and to use it safely.

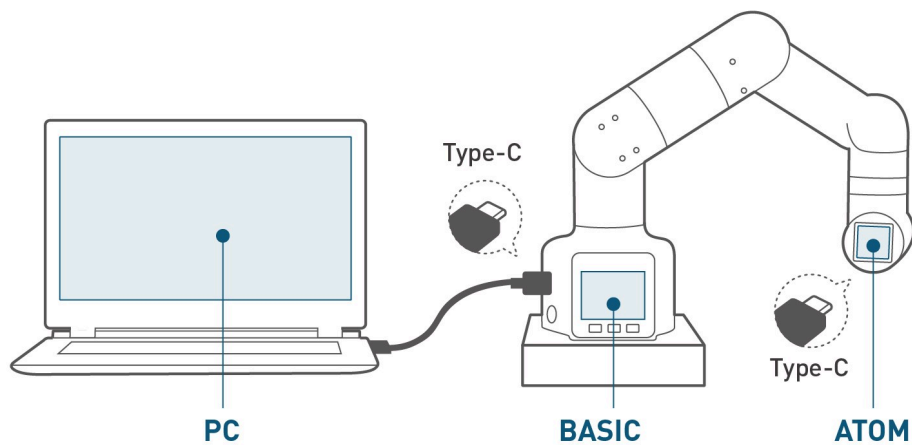
	Environment	Index	
	Temperature	-10~45℃	
	Relative humidity	20%~70%	
	Indoor and outdoor requirements	Indoor	
	other environmental requirements	Do not expose it to the sunshine Keep it away from dust, oil fume, salt, iron chips, etc Keep it away from flammable and corrosive liquids and gases It should not come into contact with water Do not apply shock, vibration, etc Keep it away from strong electromagnetic interference sources	

Step 2: Connection

Connect the power adapter as shown in the figure.



Connecting the computer



Step 3: Power supply

The power for myCobot must be supplied by an **external power supply** to provide it with sufficient power:

- **rated voltage:** 8.4-12V
- **rated current:** 3-5A
- **plug type:** DC 5.5mm x 2.1

Note: Do not supply power for myCobot only using TypeC inserted into Basic

Use the power supply officially provided by the manufacturer to avoid damage to the robot arm.

Step 4: Fixing myCobot

1 How to fix myCobot

During the movement of the robot arm, if **the bottom surface of myCobot is not connected to the table top or other bottom surfaces**, myCobot will **shake or overturn**.

There are three common ways to fix the robot arm:

- 1. Fix it on a stand with a LEGO interface by using a LEGO inserting key

We provide two types of stands: flat sucking disc and G-clamp, which can be found in [myCobot peripheral stand](#).

- Flat stand **adaptive models**: myCobot 280



- 1. Install a sucking disc on the four corners of the stand and tighten them.
- 2. Connect the flat stand and the bottom of the robot arm by using accompanying LEGO parts.
- 3. Fix the four sucking discs on a flat and smooth surface before use.

Tip: A small amount of **non-conductive liquid** can be applied under the sucking discs to fill the gap between the sucking discs and the table top for the best suction effect.



- o G-stand **adaptive models**: myCobot 280 and myPalletizer 260



- 1. Fix the stand to the edge of the table with a G-clamp.
- 2. Connect the stand and the bottom of the robot arm by using accompanying LEGO parts.

- 3. Do not use it until it becomes stable.



- 2. Pass a **screw provided for the robotic arm stand** through such stand, and fasten it to a threaded stand.
 - Sucking disc stand **adaptive models**: myCobot 320



Suitable for flat and smooth surfaces

- 1. First open the sucking discs of the self-priming pump, align the four holes in the stand bottom plate with the self-priming pump, then place the nuts in the four holes and tighten them, and finally fix the bottom plate and sucking discs.
- 2. Remove the cover at the bottom of the sucking discs, place it on a smooth table, and press the sucking discs to pump air. After pumping air from the four sucking discs, you can test whether the bottom plate has been fixed, and then tighten the nuts again.

- 3. Next, fix the robot arm to the bottom plate and align the robot arm with the groove of the bottom plate. After placing them in position, place three M6x90 screws in three holes of the robot arm, place the corresponding nuts on the bottom plate at the same position, and then tighten the screws with an Allen key to fix the robot arm on the bottom plate.



2 Connecting the screw hole of myCobot stand

The robot should be fixed on a solid stand before it can be used normally. Stand weight requirements: fixed stand or mobile stand.

Before installation, make sure that there are corresponding threaded holes on the fixed stand.

Before formal installation, confirm the following:

- The environment in which the stand is to be installed should meet the requirements in the table "Working Environment and Conditions" above.
- The installation position should not be smaller than the working range of the robot, and sufficient space should be reserved for installation, use, maintenance and repair.
- Put the stand in a suitable position.
- Installation-related tools, such as screws, wrenches, etc. should be ready.

After confirming the above points, move the robot onto the stand mounting table, adjust the robot position, and align the fixing holes of the robot stand with the holes in the table. After aligning the holes, align the screws with the holes and tighten them.

Notice: When adjusting the robot position on the table, don't push and pull the robot on the table directly as much as possible to avoid scratches. When manually moving the robot, don't apply external force to the vulnerable parts of the body so as to avoid unnecessary damage to the robot.

Video Tutorial

[Opening the box of myCobot quickly](#) [Opening the box of myCobot Pro quickly](#)

Introduction to PI version robots

- **Supported device models**

Robot	Introduction
myPalletizer 260 PI	Introduction
mechArm 270 PI	Introduction
myCobot 280 PI	Introduction
myCobot 320 PI	Introduction
myBuddy 280	Introduction

- **PI device function description**

- PI version robots is a joint product developed by Elephant Robotics and **Raspberry Pi**. Adopting the **Raspberry PI 4B** as its core processor, and reserving the origin hardware interface of RaspBerry PI 4B, the robotic arm meets the application needs of **Linux system** with one-in-all structure for portable robot development. Featuring built-in **Ubuntu 18.04. system** as well as development environments like **Python**, **ROS** and **myBlockly**, it can be developed after connecting with displayer without matching with PC.
- PI version robotic arms is embedded with Raspberry PI 4B, 1.5GHz 4-core microprocessor, runs Debian/Ubuntu platform, supports 4 USB channels, 2 HDMI channels, standardized GPIO interface, TF card is removable

- **PI robots are computer hosts**

- The essence of the PI version robotic arms is a development board with an independent system, which can be seen as a miniature computer host. The communication between the host and the host cannot be simply constituted by a wire. It can only be connected to an independent monitor, and equipped with power supply, mouse and keyboard, and can be developed and operated after entering the built-in system of the development board

- **The difference between PI robots and other robots**

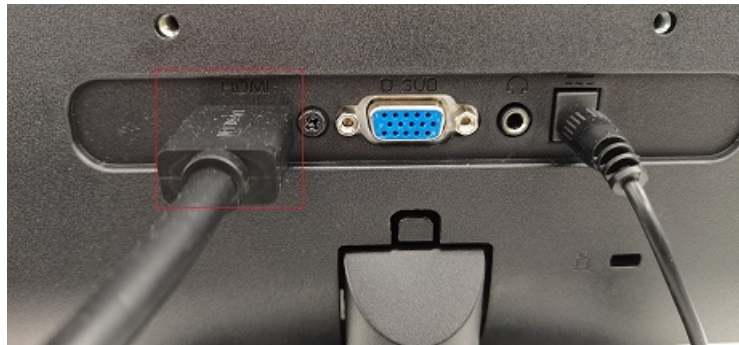
Detailed differences between the M5 and PI versions can be found [here](#)

Introduction to PI version robots

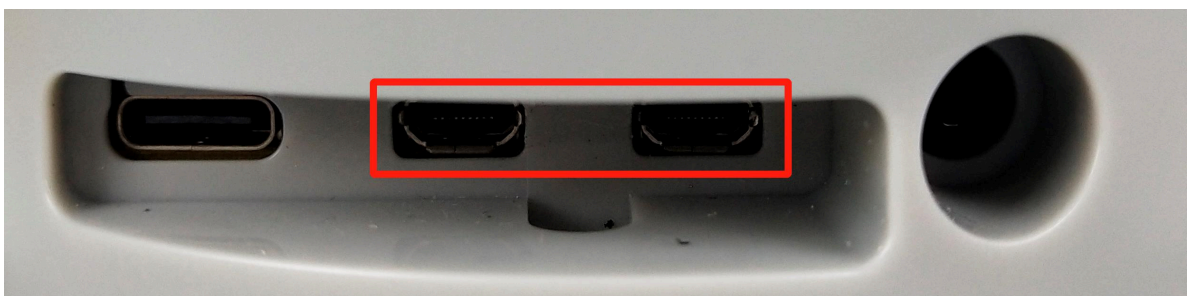
1.1 Start to use

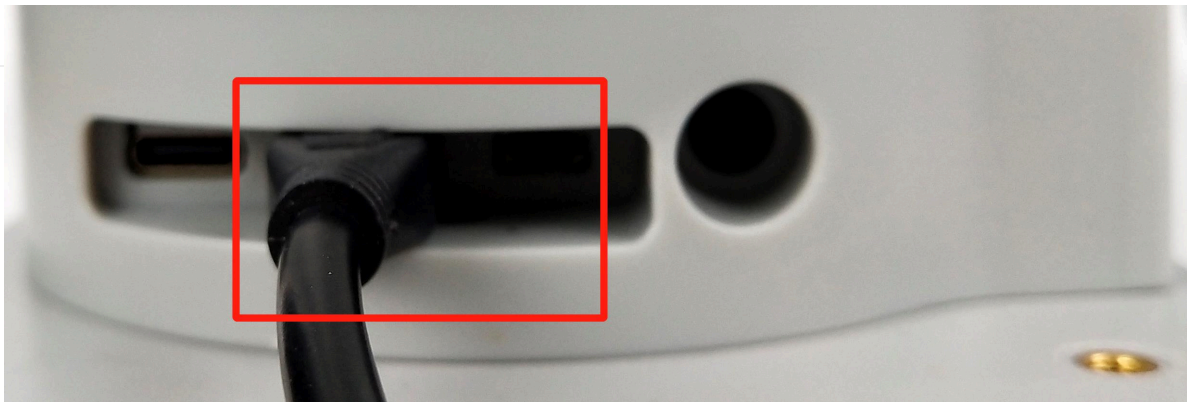
- **Connect devices**

- PI version robotic arms does not need to be equipped with a PC, laptop and other equipment, and can be connected to the display for application development (**Tips△: Use the delivered HDMI cable to connect the monitor and use the built-in system for development**)
- Plug the HDMI cable into the HDMI port of the monitor.



- Plug the other end into the HDMI port of the robotic arm.





- **SD card description**

- 32G TF card, built-in Ubuntu20.04 system, had installed **myStudio**, **myBlockly**, adapted **python ROS** develop env.

- **Optional items**

- A network (Ethernet) cable to connect your Raspberry Pi to your local network and the Internet.
- If you aren't using an HDMI monitor with speakers you might also need some form of sound hardware. Audio can be played through speakers or headphones by connecting them to the AV jack (not available on the Raspberry Pi 400). However speakers must have their own amplification since the output from your Raspberry Pi is not powerful enough to drive them directly.

- **Troubleshooting**

- Make sure you are using a good quality power supply; we recommend using an official power supply.
- Make sure you are using a good quality power supply; we recommend using an official Raspberry Pi power supply.
- You can get help with using the robot on our gitbook

1.2 Updates of the system

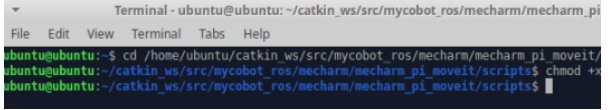
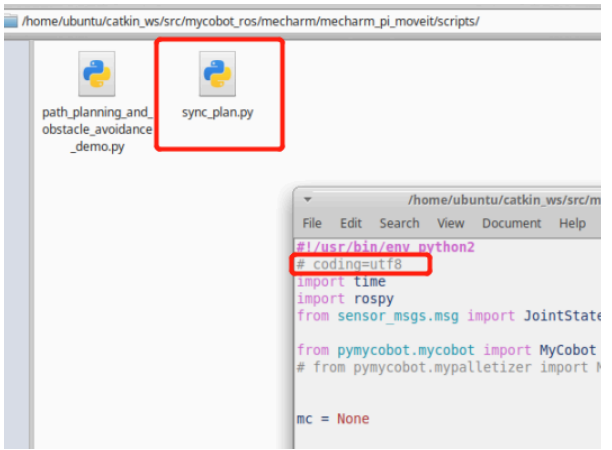
- **What is Mirroring**

- Mirroring is a form of file saving. It refers to the fact that data saved in one disc also exists in another disc without any distortion. Mirroring files are often saved as BIN, IMG, TAO, DAO, FCD. It is similar to ZIP packages, which make a series of files into one single file according to certain formats to meet users demands. The most fundamental function of mirroring is that it can be identified by a software immediately and recorded on disc. Generally, mirroring files can be extended to cover more information such as system files. Thus, mirroring files can contain the information of even a hardware. The most typical software for creating mirroring files is Ghost, featuring recording function to save information on a disc.

- **Download system image files**

- Download links

Product	Version	Link	SHA256 Hash
AI Kit 280	ubuntu 18.04	Download	d44439be351a52decdb4470cb623a032047e223ffce734
myCobot 280 PI	ubuntu 18.04	Download	04e40af5b637ec003a8b23ef9012e353361fd336db4e17c
	ubuntu 20.04	Download	ce666e6c1047c512fe6b270336d472e48f231be1280872
myCobot 280 JetsonNano	ubuntu 18.04	Download	2f1e40c1480b077bcc83abd3b79ac175f25d21e9cc344a
myCobot 320 PI	ubuntu 18.04	Download	bc2ed6ef8d51a885f45379392b71e35420638a427d5b4b
	ubuntu 20.04	Download	c95633bfd49246254f2be4783c6a91a152124222191579

	ubuntu 18.04	Download	9af1fcbf9c608eda269dc395a8d68ea0a270008a88ec8ec
			Solution: Execute moveit and the following prompt will appear, ind permission has been granted: <pre>ubuntu@ubuntu:~\$ roslaunch mecharm_pi moveit sync_plan.py [roslaunch] Couldn't find executable named sync_plan.py below /home/ubuntu/catkin_ws/src/mycobot_ros/mecharm/mecharm_pi_moveit/scripts/ [roslaunch] Found the following, but they're either not files, [roslaunch] or not executable: [roslaunch] /home/ubuntu/catkin_ws/src/mycobot_ros/mecharm/mecharm_pi_moveit/scripts/sync_plan.py</pre> Enter the path shown in the figure and give the py file ex  Execute moveit and the following prompt will appear, ind encoding format in the code is wrong: <pre>ubuntu@ubuntu:~\$ roslaunch mecharm_pi moveit sync_plan.py _port:=/dev/ File "/home/ubuntu/catkin_ws/src/mycobot_ros/mecharm/mecharm_pi_moveit/scripts/sync_plan.py", line 36 SyntaxError: Non-ASCII character '\xe5' in file /home/ubuntu/catkin_ws/src/mycobot_ros/mecharm/mecharm_pi_moveit/scripts/sync_plan.py on line 36, but no encoding n.org/dev/peps/pep-0263/ for details</pre> Enter the path shown in the figure, open the py file and e #coding=utf8 and save it 
mechArm 270	Mirror Known Issues	moveit function abnormal	
myPalletizer 260	ubuntu 18.04	Download	f6fe999519146428e4c60960b242f647ae5c73c704852dc
	ubuntu 20.04	-	-

myCobot Pro 600	Raspberry Debian	Download	2e73aaa153bddbf0a49d18669a254b27403f17f8e989c0f
myBuddy 280	ubuntu 20.04	Download	2b5452f665bcb999faf1727b2103dc1e5745705f5706728
myAGV	ubuntu 18.04	Download	bedad7d9769cb69380c6a4b9742ba7aefc21db41ab2391
marsCat	-	-	-

- **How to update the system**

Step 1: Unzip the package and a file of image style appears.

名称	修改日期	类型
 myCobot_280_ubuntu_V20221101_2_20.04...	2022/11/2 16:59	光盘映像文件 image file

Step 2: Download Win32DiskImager.

Go to [Win32DiskImager](#) to download.



LibreOffice is a powerful open source office suite.

[Download LibreOffice for Free](#)

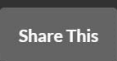
Home / Browse / System / Storage / Win32 Disk Imager



Win32 Disk Imager

A Windows tool for writing images to USB sticks or SD/CF cards
Brought to you by: [gruemaster](#), [tuxinator2009](#)

★★★★☆ 123 Reviews Downloads: 27,922 This Week Last Update: 2018-06-07

Windows

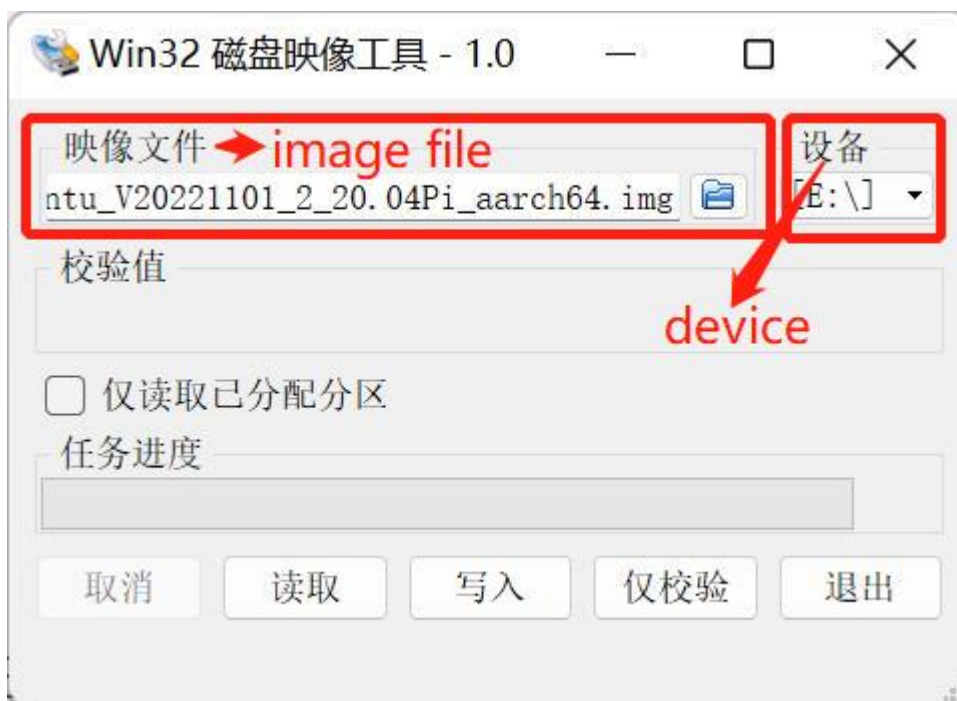
Step 3: Remove SD card from the pedestal, and then insert the SD card into PC.

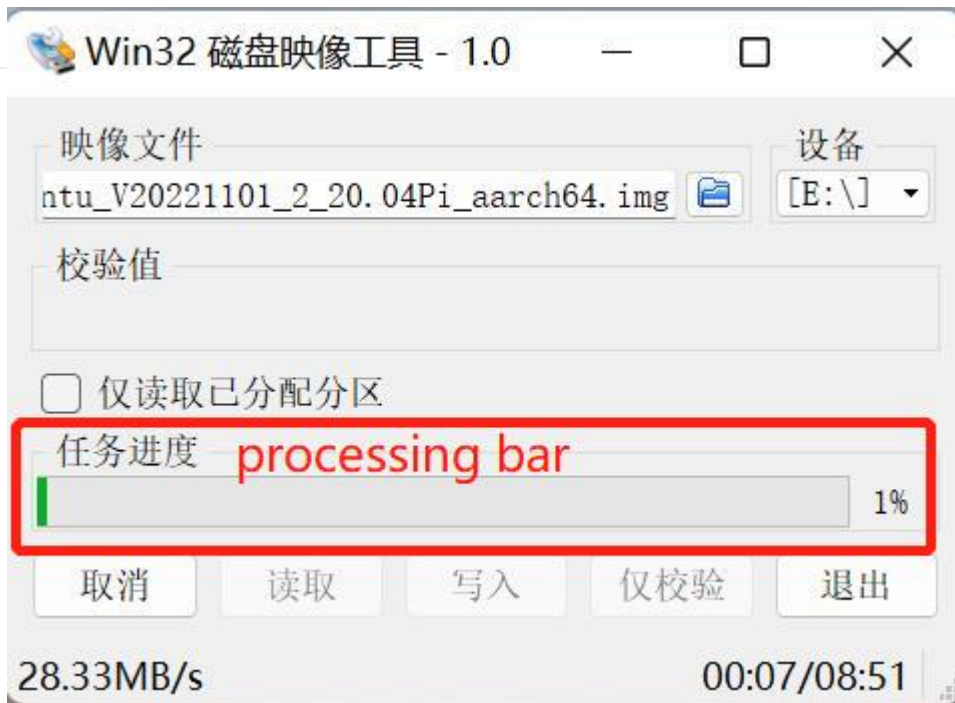


Step 4: Open Win32DiskImager.

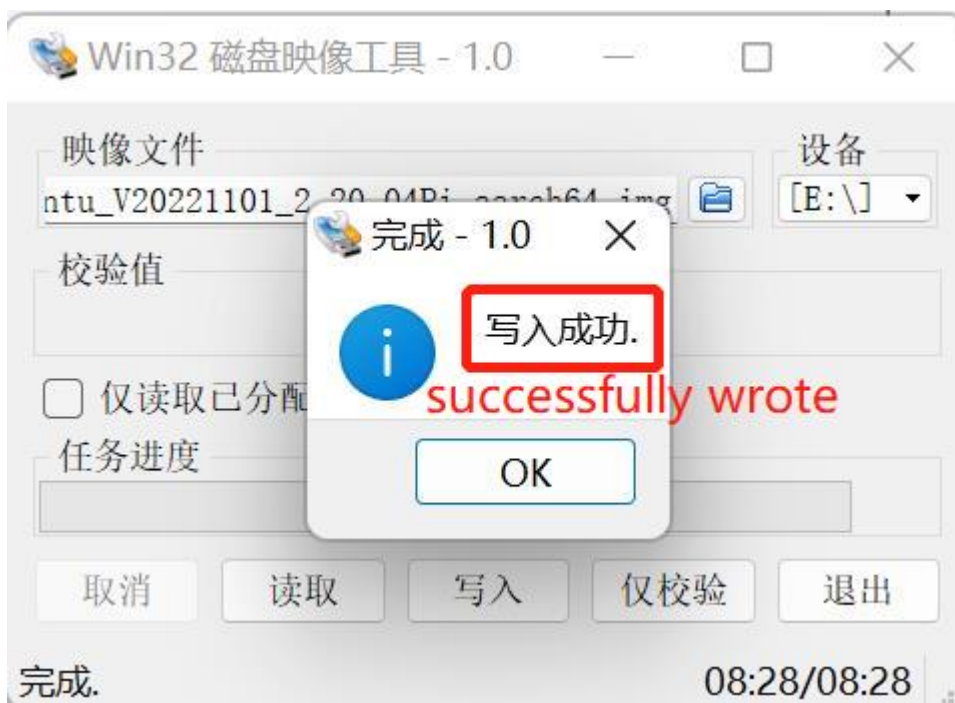


Step 5: Select the software and device (E disc) and then write the software into PC.





Step 6: Successfully processed.



Description of basic system functions

2.1 Robot system introduction

- **System**
 - Ubuntu is the most widely used linux operating system for personal desktop operating systems. For beginners, familiarity with the linux environment or some embedded hardware operating system is a good choice. The official ubuntu website has also released a dedicated operating system for the Raspberry PI.



- **Function introduction**
 - **myStudio**: Firmware burning software for updating and burning new firmware.
- **myBlockly**: Graphical programming software, can be directly by dragging building blocks to form running code, control the robot arm
- **ROS1 Shell**: Directly enter the compiled ROS1 environment, you can directly enter the corresponding instructions, run the corresponding ROS1 code
- **ROS2 Shell**: Directly enter the compiled ROS2 environment, you can directly enter the corresponding instructions, run the corresponding ROS2 code
- **Github-ElephantRobotics**: Elephant robotics official open source code repository
- **Home-ElephantRobotics**: Elephant robotics website
- **UserManual - CN/EN**: Robot use manual, contains everything about robot control
- **WIFI_ON/OFF**: WiFi switch, click to turn on/off the WiFi function
- **HotSpot_ON/OFF**: Hotspot switch, click to turn on/off the hotspot function, the hotspot name is **ElephantRobotics_AP_XXXX**
- **Language Support**: System language setting, click to enter the system language setting interface
- **System details documentation link**
 - [myPalletizer 260 PI Development environment detailed introduction](#)
 - [mechArm270 PI Development environment detailed introduction](#)

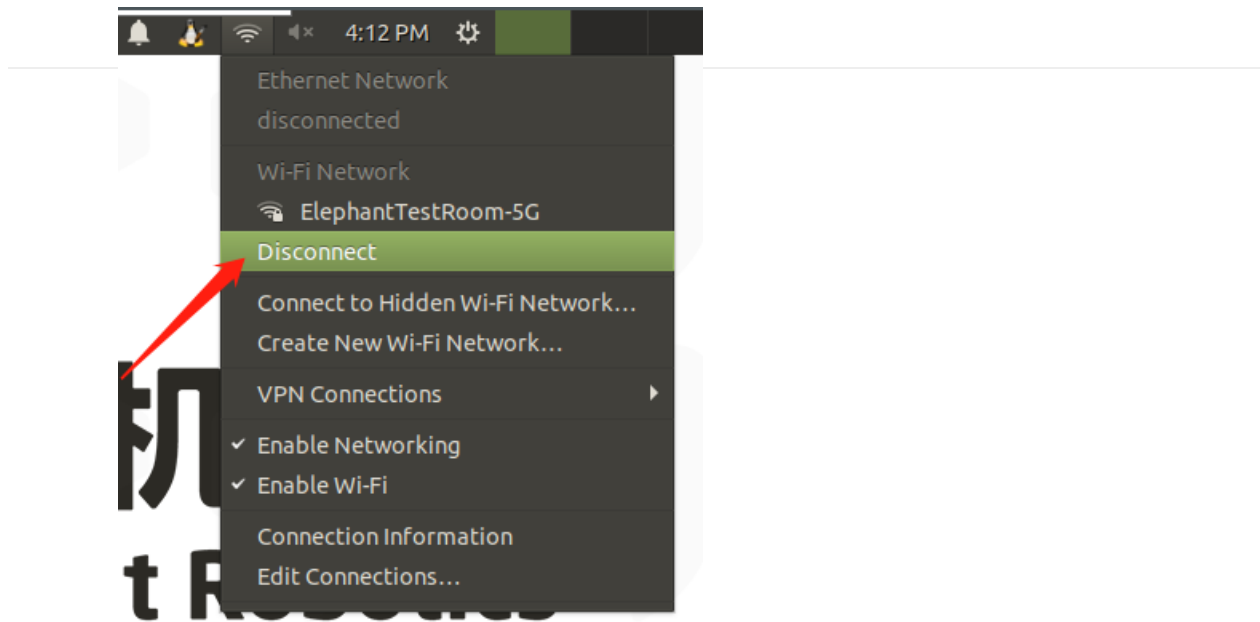
- [myCobot280 PI Development environment detailed introduction](#)
 - [myCobot320 PI Development environment detailed introduction](#)
-

2.2 System password description

- **account password & VNC password & SSH password & root account password**
 - Password: **Elephant**
- **How to change password**
 - change account password
 - Use the shortcut key `ctrl + alt + T` to open the terminal
 - Enter `passwd`
 - Enter the new password twice
 - change vnc password
 - Use the shortcut key `ctrl + alt + T` to open the terminal
 - Enter `vncpasswd`
 - Enter the new password twice
 - change ssh password
 - For SSH remote connection, the password of the administrator account is entered. You do not need to change the password separately
 - change root account password
 - Use the shortcut key `ctrl + alt + T` to open the terminal
 - Enter `sudo passwd`
 - Enter the new password twice

2.3 VNC

- **VNC Introduction**
 - Is a remote control software, generally used to remotely solve computer problems or software debugging
- **VNC Port**
 - When the robot arm and PC are connected to the same WiFi, the IP address of the robot arm is the port
- **Connect VNC**
 - There are two wireless connection modes. The first mode requires an external monitor to do some operations on the system. The specific steps are as follows: Click **“Disconnect”**, disconnect the default hotspot connection



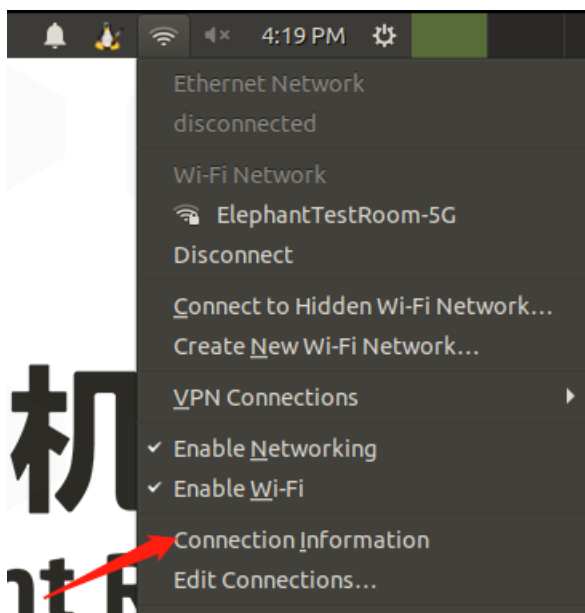
Click **"Enable Wi-Fi"** , and the currently available WiFi will appear



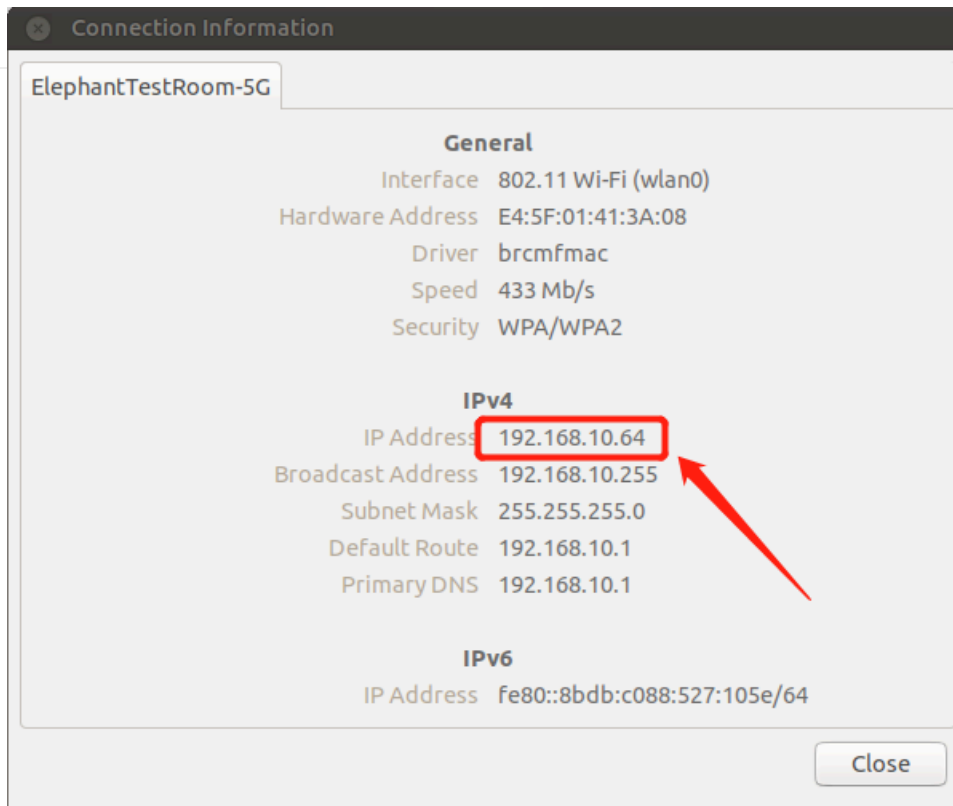
Click on the WiFi you need to connect to, enter the password



After connect successfully, click "**Connection Information**" to check IP address of the robot



As shown in the example, "**192.168.10.64**" is the current IP address of the robot

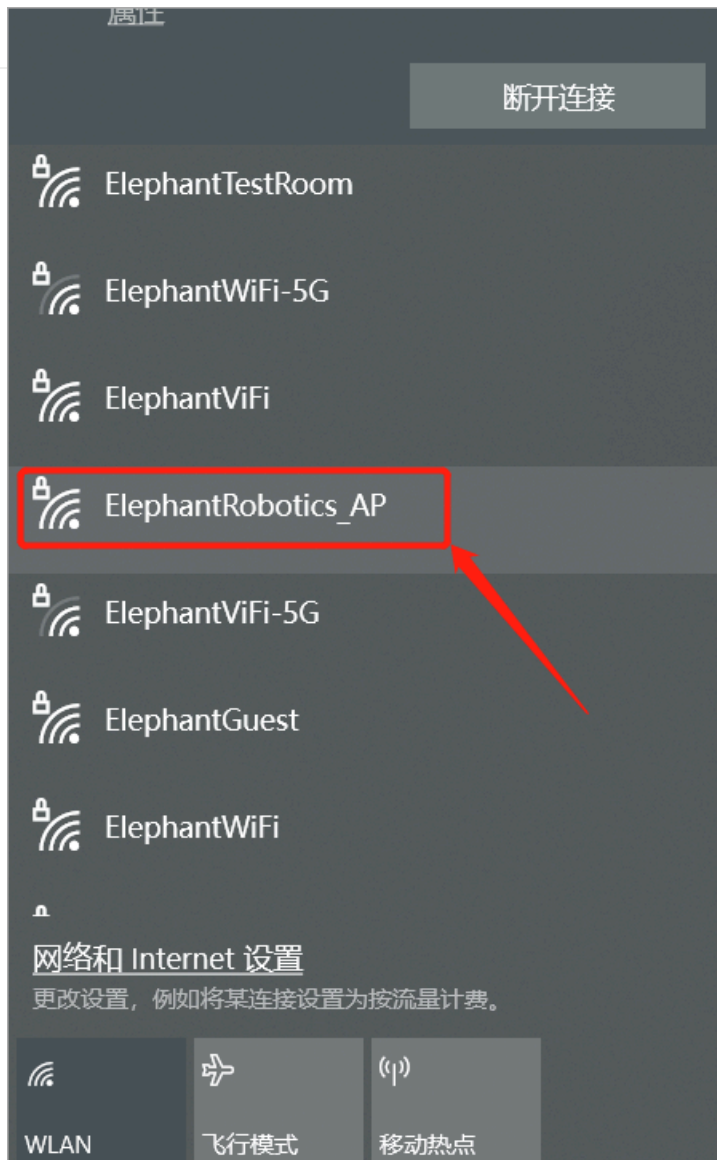


Connect your PC and the robot to the same WiFi, open the VNC viewer, enter the IP address (examples : input **192.168.10.64**) , enter the password **Elephant**, The user name is not specified by default. The following is an example of a successful connection :



- o The second method doesn't need to connect the monitor, directly connect the Ubuntu system hotspot with a PC for remote control, but this connection method does not have the function of Internet surfing, and can only remotely control the robot arm system. The specific steps are as follows:

Connect tp the hotspot **ElephantRobotics_AP_XXXX**, enter the password **Elephant**



Open VNC viewer, input IP address **10.42.0.1** , enter, and then enter the password **Elephant**, The user name is not specified by default. The following is an example of a successful connection:



- **How can I improve connection fluency**

- The fluency of the remote connection depends on the fluency of the connected WiFi. It is recommended to connect to a stable WiFi for remote control

2.4 SSH

- **SSH Introduction**

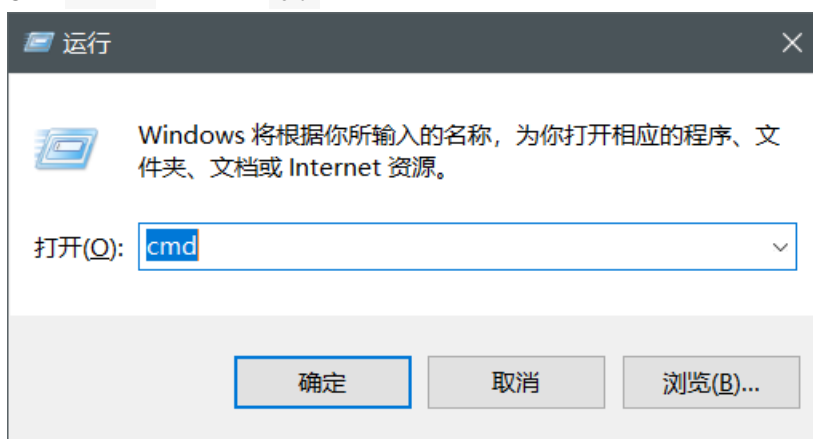
- SSH is a network protocol used for encrypted logins between computers. If a user logs in from a local computer to another remote computer using SSH, we can assume that this login is secure, even if it is intercepted in the middle, the password will not be disclosed.

- **SSH Port**

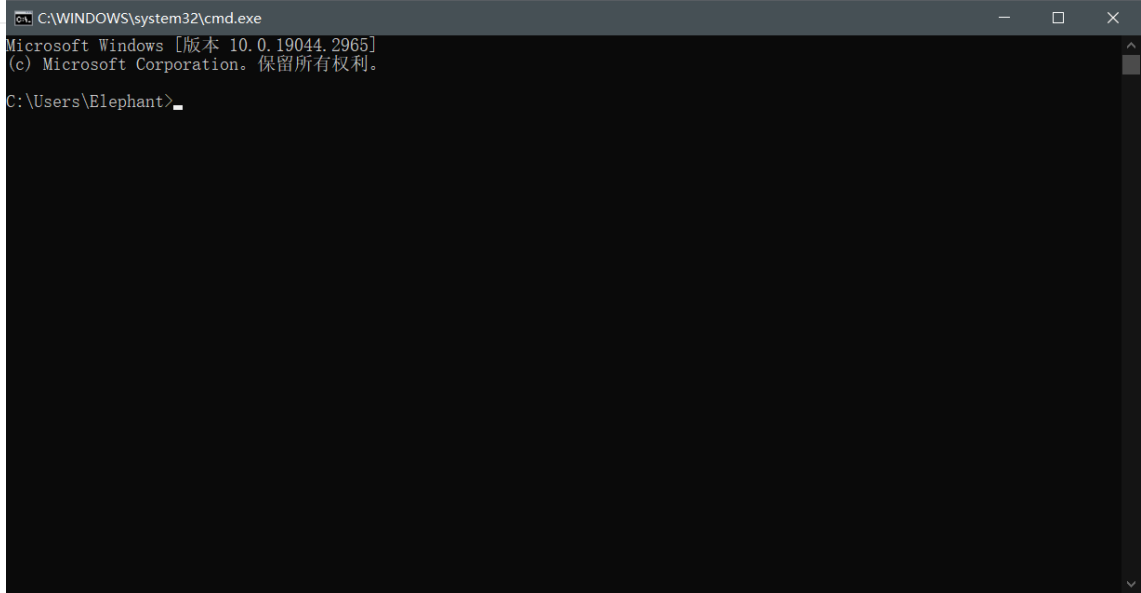
- The default port number is 22

- **SSH Connect**

- Confirm the IP address of the robot by following instructions in **2.3 VNC**
- Click `win + R` and enter `cmd`



- o After input, click OK to open the shell interface



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Elephant>
```

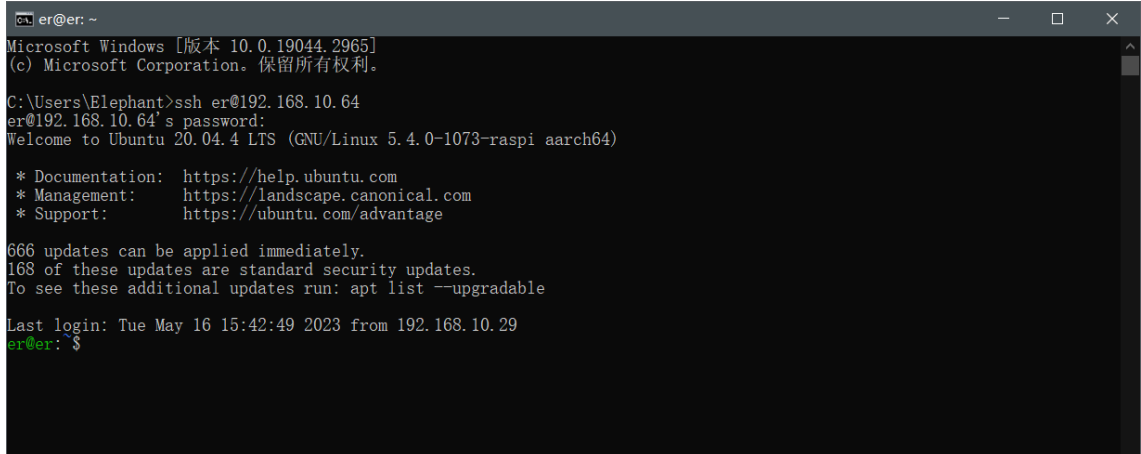
- o Enter `ssh er@IP address` , then press `Enter` (the IP address is mainly displayed by the robot arm, the figure is only an example)



```
C:\WINDOWS\system32\cmd.exe - ssh er@192.168.10.64
Microsoft Windows [版本 10.0.19044.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Elephant>ssh er@192.168.10.64
er@192.168.10.64's password:
```

- o Enter password `Elephant`



```
er@er: ~
Microsoft Windows [版本 10.0.19044.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Elephant>ssh er@192.168.10.64
er@192.168.10.64's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-1073-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

666 updates can be applied immediately.
168 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Tue May 16 15:42:49 2023 from 192.168.10.29
er@er: $
```

- o As shown in the above figure, the robot arm has been successfully connected by ssh

- **How can I improve connection fluency**

- o The fluency of the remote connection depends on the fluency of the connected WiFi. It is recommended to connect to a stable WiFi for remote control

2.5 Network configuration

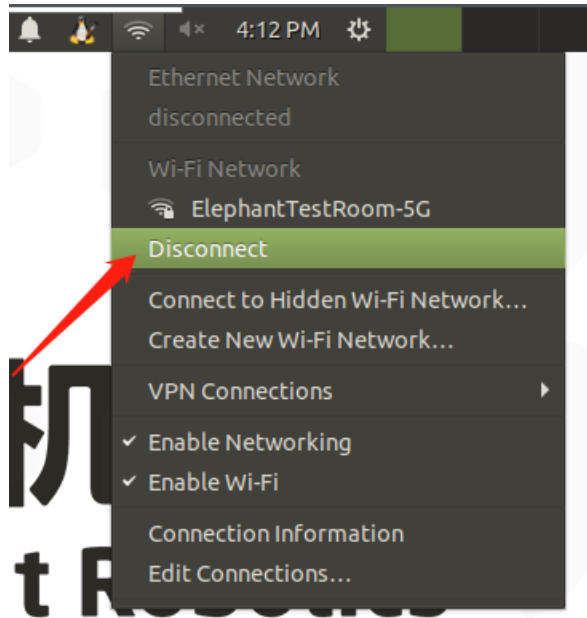
- **Default AP usage**

- o Power on the robot, by default, the system will connect to the hotspot generated by the PI itself, the hotspot name is **ElephantRobotics_AP_XXXX**, current IP address **10.42.0.1**, this hot spot does not have the function of web surfing, and the transmission rate and information is limited, so there will be some

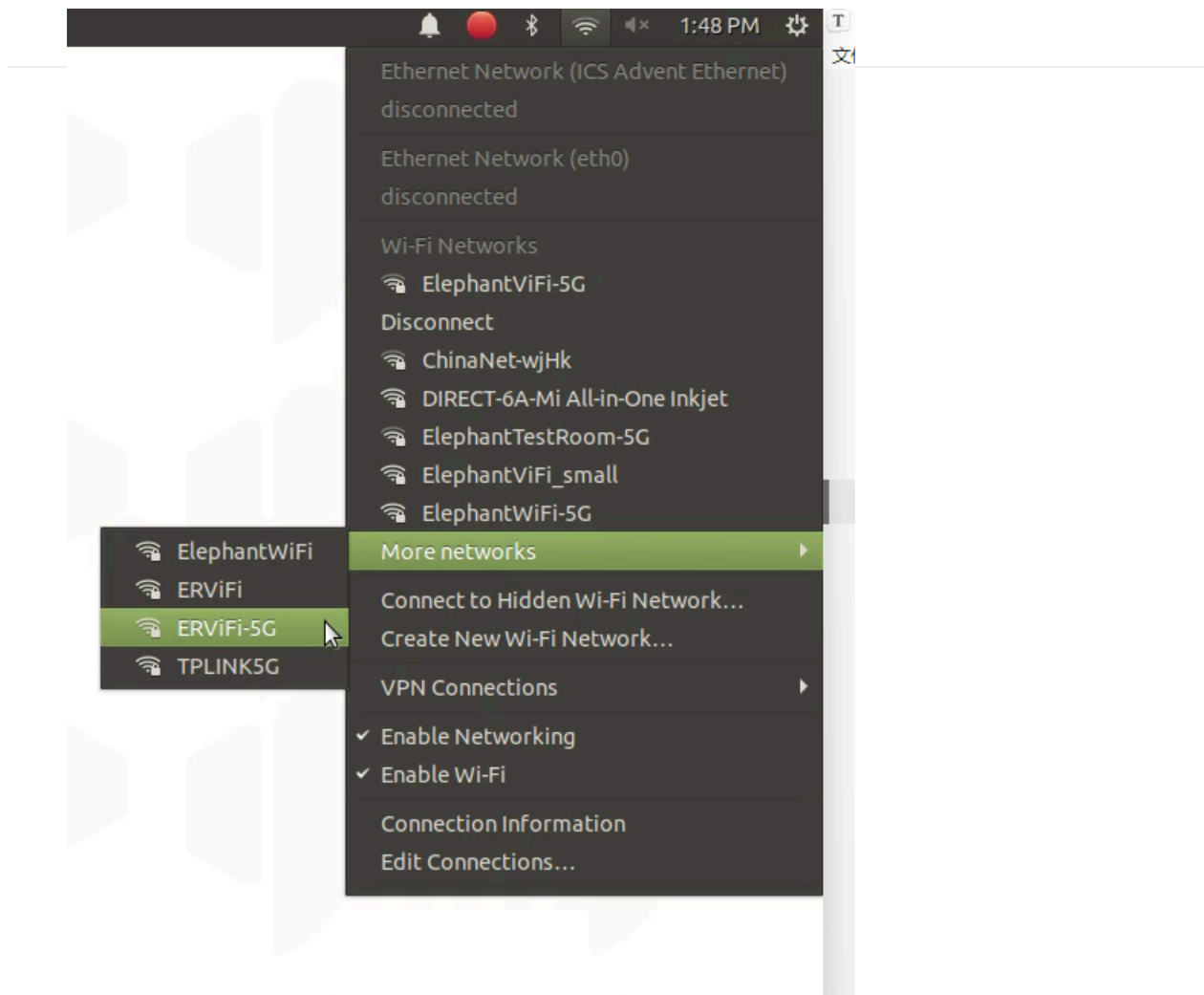
distortion and color difference in the final imaging, and there will be a delay in communication transmission, which is a normal phenomenon

- **Connect to WLAN**

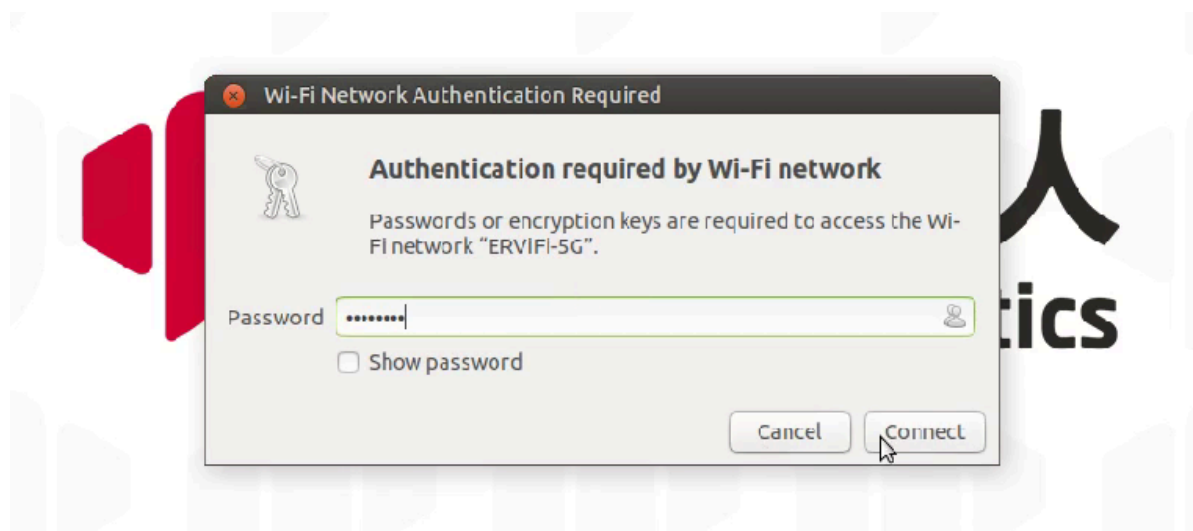
Click **"Disconnect"** to turn off default hotspot



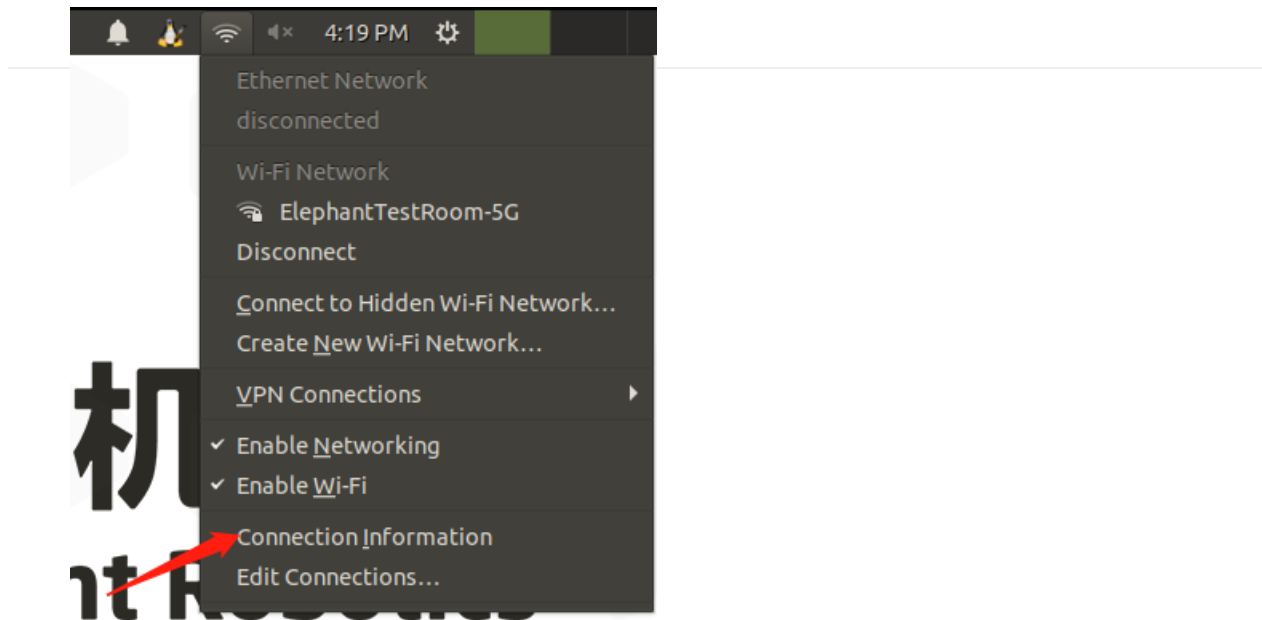
Click **"Enable Wi-Fi"** , wait for the currently available WiFi to be displayed



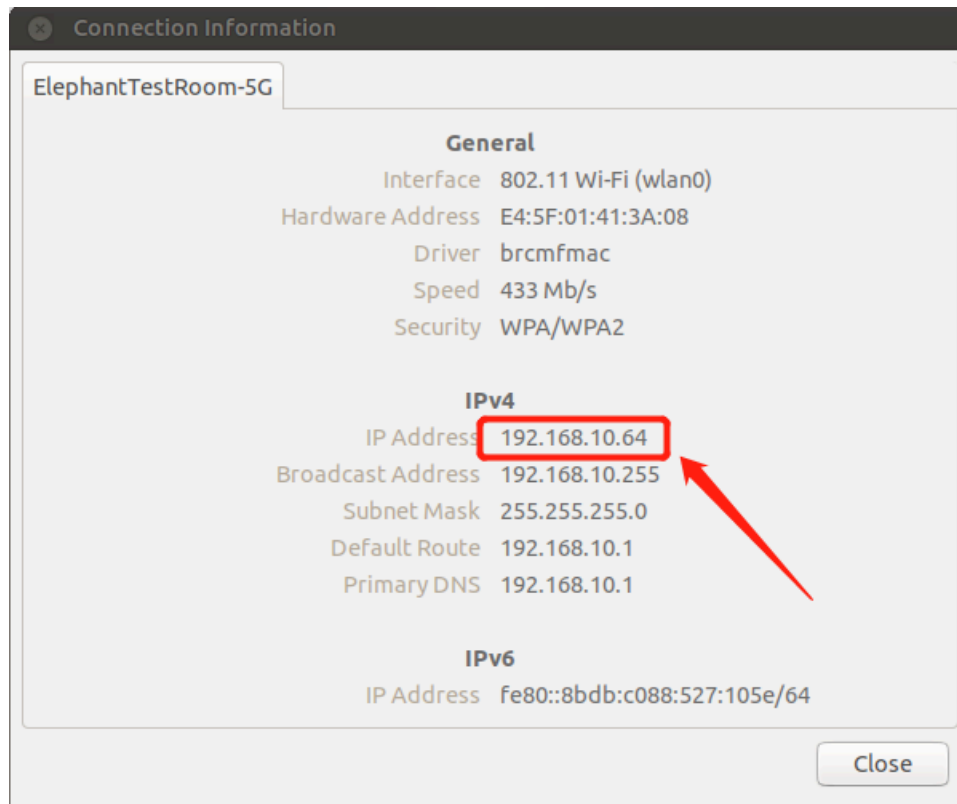
Click the WiFi you want to connect to and enter the WiFi password



After connect successfully, click "Connection Information" to check IP address

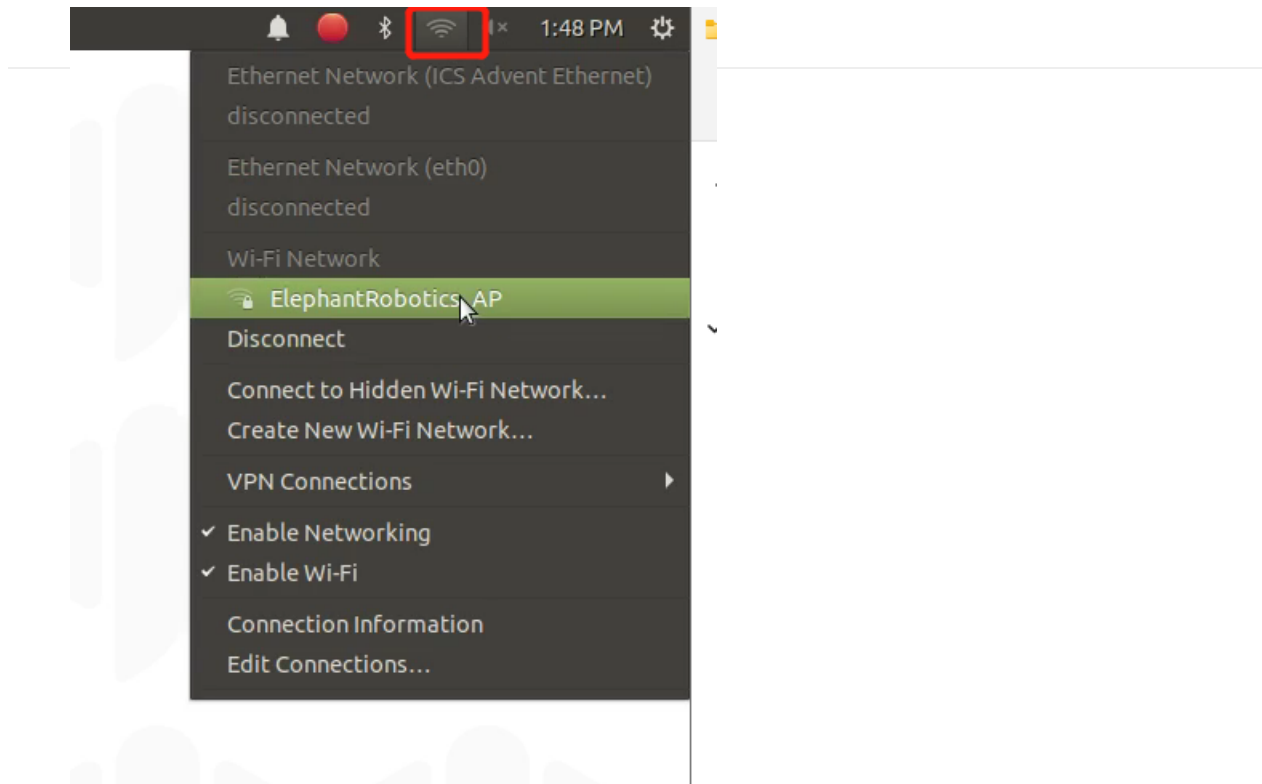


As shown in the example, “192.168.10.64”, It is the current IP address of the robot

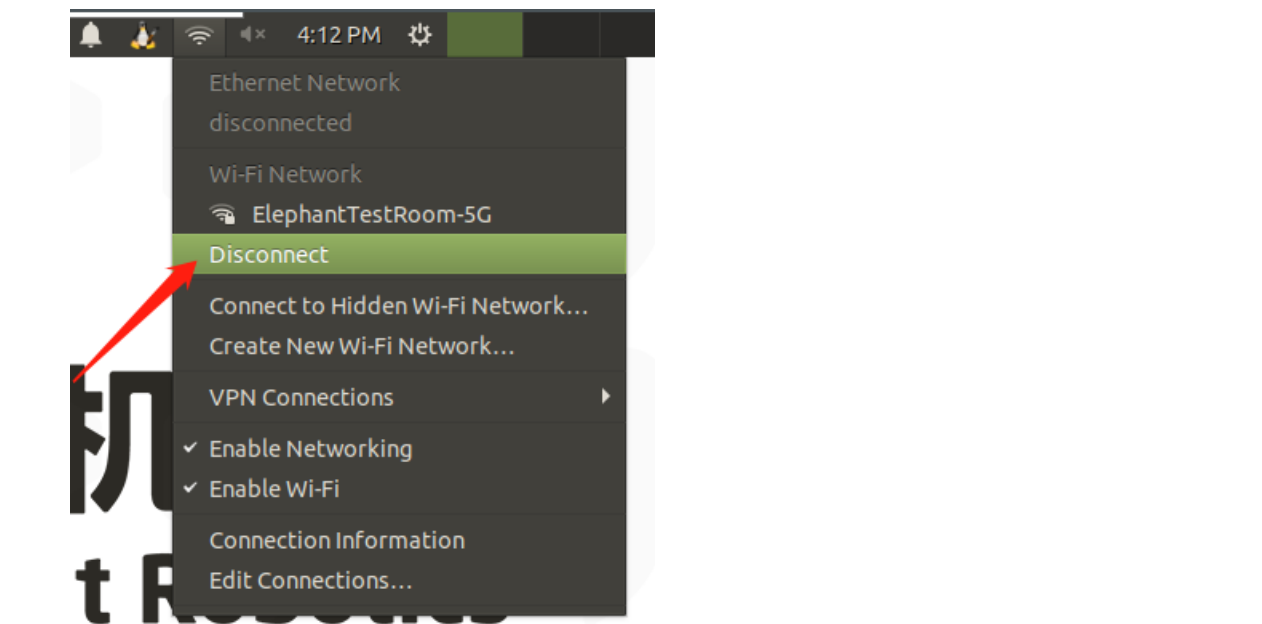


- **Wired network connection**

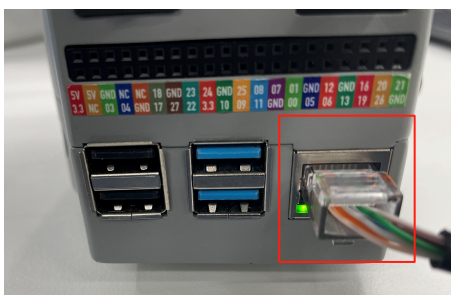
Power on the robot, it is connected to the hotspot configured by the system by default:
ElephantRobotics_AP_XXXX



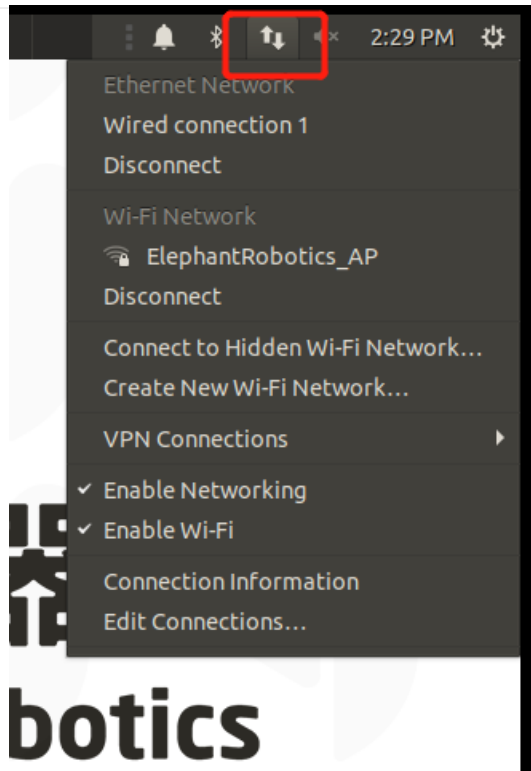
Click **“Disconnect”**, disconnect from the default hotspot



Connect the network cable to the network port of the robot



Connect the Internet cable to the network port of the robot



- **How to set default IP address**

Use shortcut keys `ctrl + alt + T` to open the terminal, enter `sudo vim /etc/netplan/01-network-manager-all.yaml` , modify 01-network-manager-all.yaml file as follows:

```
er@er: ~  
File Edit View Search Terminal Help  
# Let NetworkManager manage all devices on this system  
network:  
  version: 2  
  renderer: NetworkManager  
  ethernet:  
    eno1:  
      addresses: [192.168.10.127/24]  
      dhcp4: false  
      optional: true  
      gateway4: 192.168.10.1  
      nameservers:  
        addresses: [192.168.10.1]
```

After modification, enter `sudo netplan apply` to make the configuration take effect.

- **How to assign IP addresses automatically**

When the system is connected to WiFi, the IP address is automatically assigned, If you want to change the fixed IP address to automatically assign an IP address, modify `/etc/netplan/01-network-manager-all.yaml` file follows:

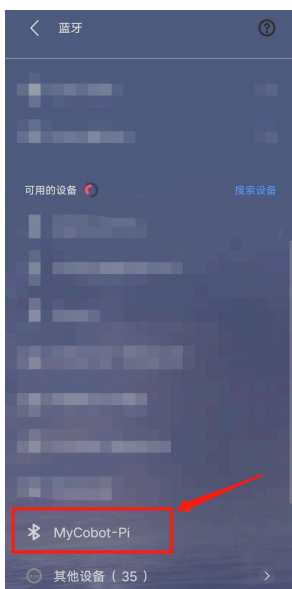
```
er@er: ~  
File Edit View Search Terminal Help  
# Let NetworkManager manage all devices on this system  
network:  
  version: 2  
  renderer: NetworkManager
```

<etc/netplan/01-network-manager-all.yaml" 12L, 352C 12,29 All

After modification, enter `sudo netplan apply` to make the configuration take effect.

2.6 BT configuration

- The Bluetooth of the system is turned on by default. Use the PC/Phone directly , and you can search the robot's BT. The default name of the BT is **MyCobot-Pi**



- Transfers files from PC/Phone to the robot system
 - Select the file you want to transfer
 - Operate in the robot system and click **Accept** to receive file



- o Wait for the BT transfer to complete



- o Can check received files in **/home/er/Downloads** folder

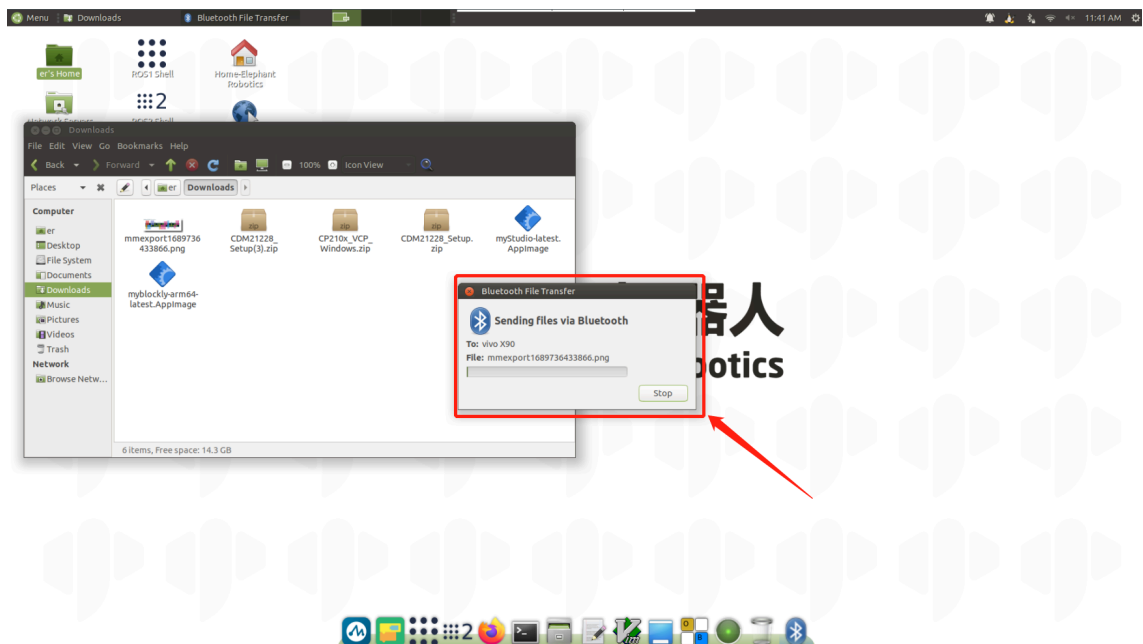
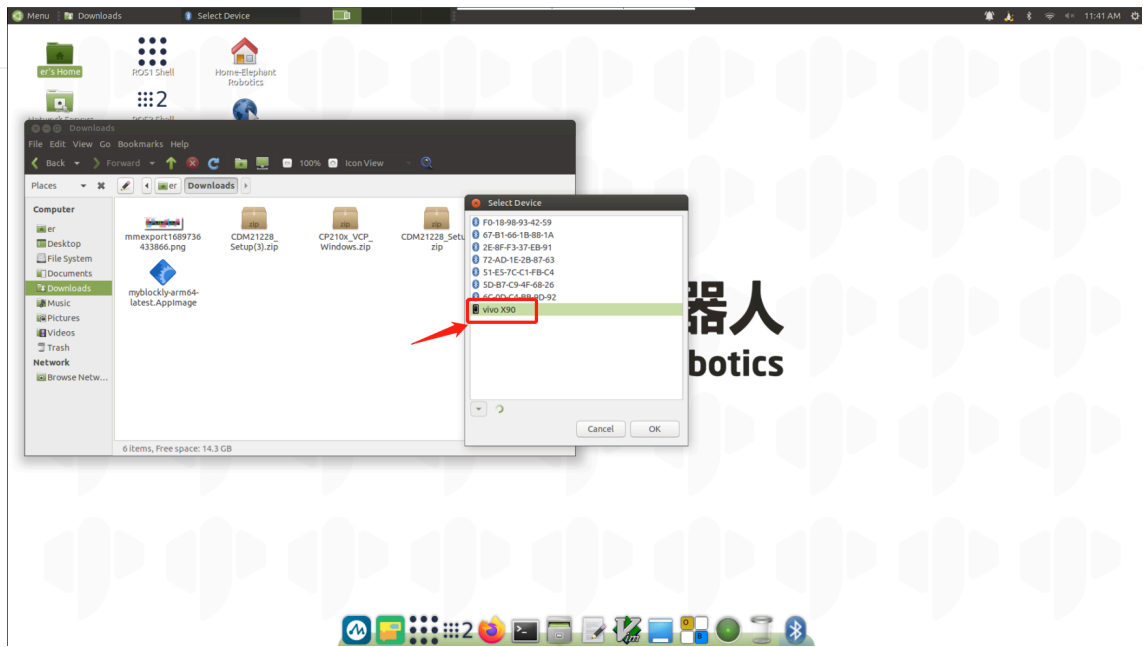


- Transfers files from the robot system to PC/Phone
 - Click BT icon, select **Send Files to Device**

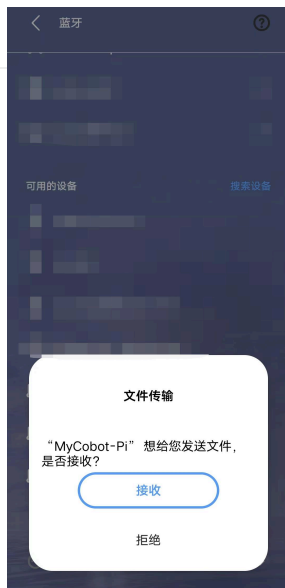


- Choose PC/Phone

1 myCobot Series 6-Axis Collaborative Robotic Arm



- o On PC/ mobile phone, allow the device to receive files



2.7 Language configuration

- How to change language

Click **Language Support**, drag the language you want to the top and restart the system



- How to download language

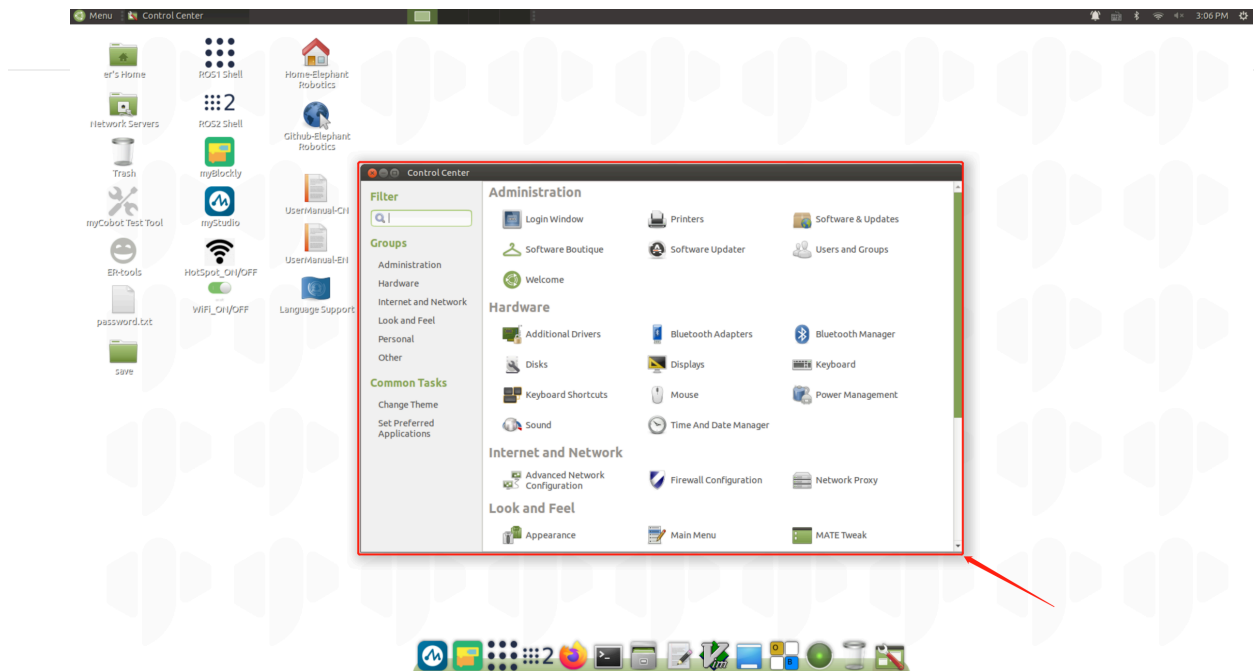
Click **Language Support**, choose language, click to download, enter password **Elephant**



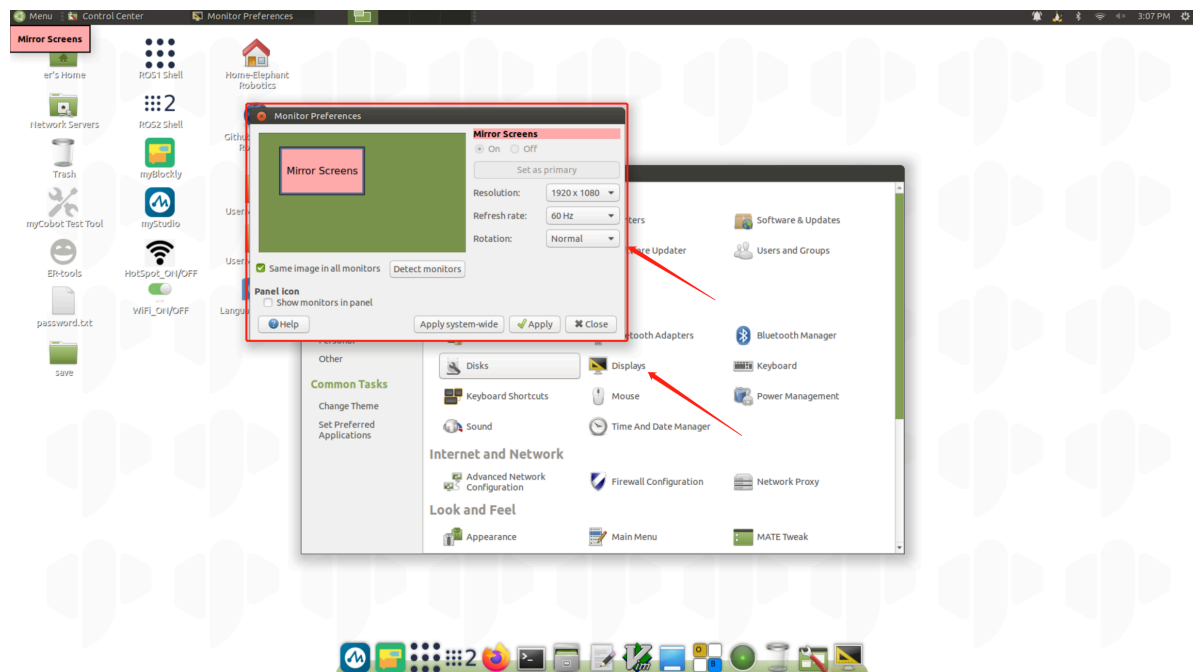
2.8 System resolution switch

- Click the icon in the upper right corner of the screen and select **System Settings** to enter the control panel



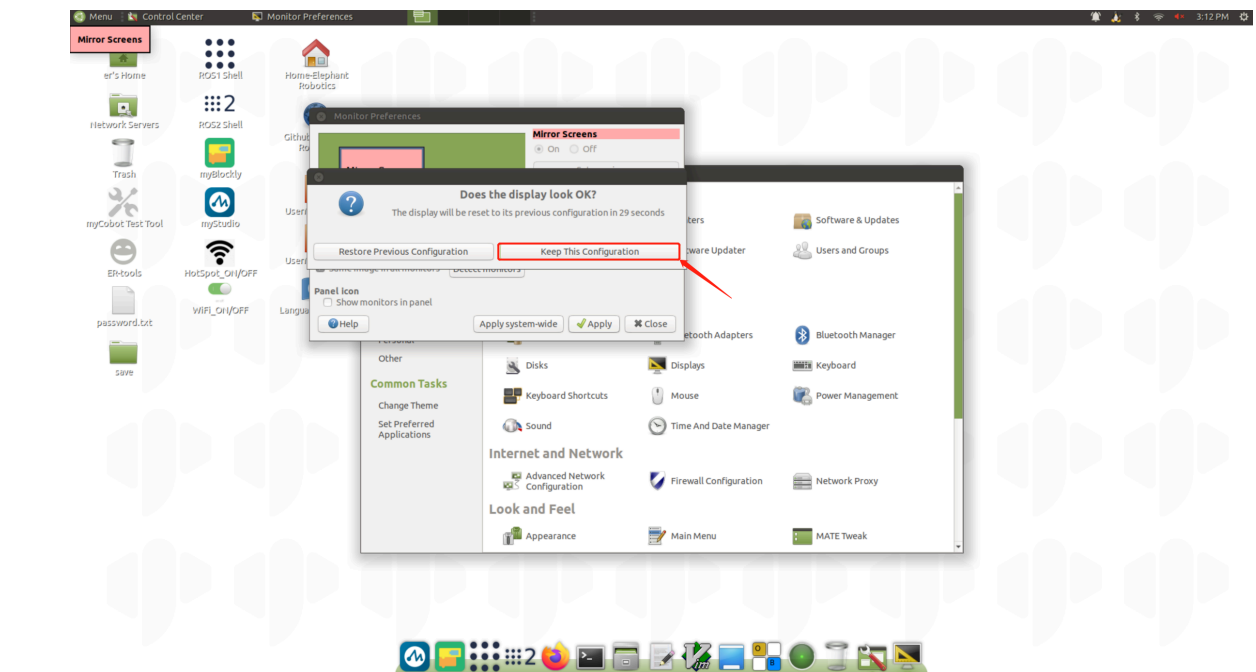
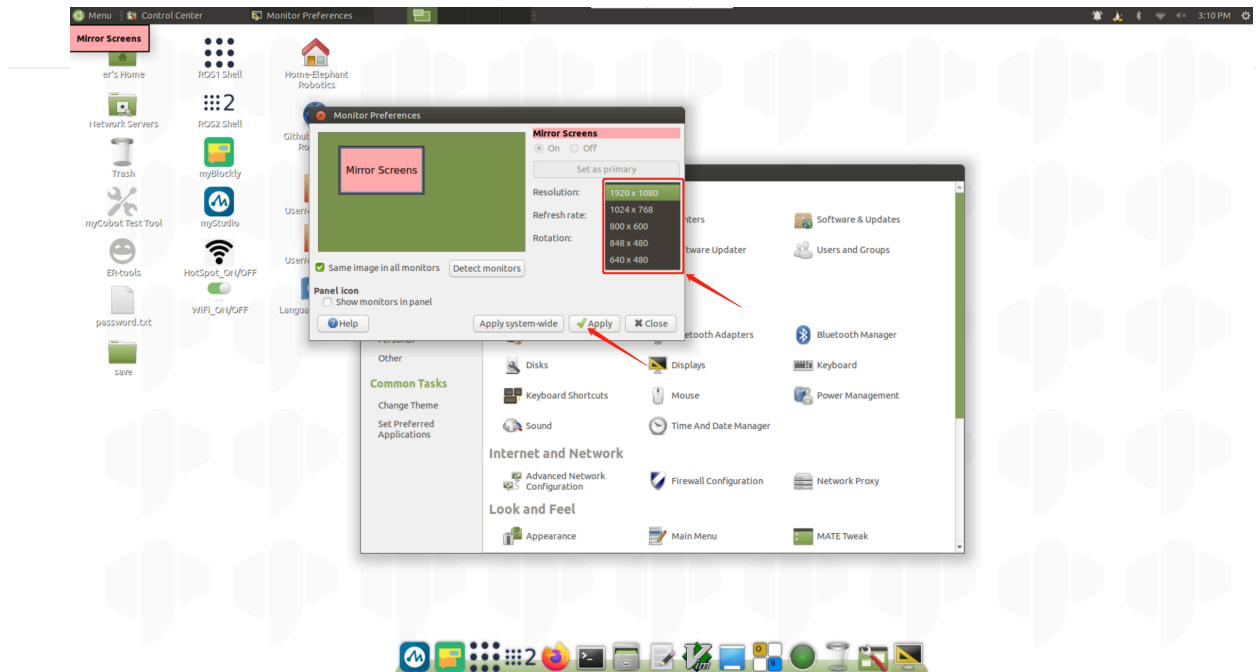


- Choose **Display**



- Toggle the selected resolution and click **Apply** to check and click **Keep this Configuration** to save the configuration

1 myCobot Series 6-Axis Collaborative Robotic Arm



2.9 python

- **Python introduction**

Built-in installation **Python3.8**, no need to install it yourself

Installed libraries:

Package	Version
pymycobot	3.1.5
pyserial	3.5
numpy	1.23.5
opencv-contrib-python	4.7.0.72
rospkg	1.4.0
rospkg-modules	1.4.0

- **Try to program**

If you are new to the python programming language, you can follow the following video to program



You can try this code in the following input fields:

```
print ("Hello World!")
```

- **Run demo**

Specific case codes can be viewed [Python](#), just copy the code in the cases and use it

- **Corresponding chapter case**

[1 Joint control](#)

[2 Coord control](#)

[3 IO control](#)

[4 Gripper control](#)

[5 TCP/IP control](#)

[6 Handle control](#)

[7 Example and videos](#)

Hardware interface usage

3.1 40PIN GPIO

- **Introduction**
 - GPIO: General-purpose input/output
 - At present, all our Raspberry PI motherboards have a 40-pin GPIO head
- **Voltages**
 - Two 5V pins and two 3.3V pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3.3V pins, meaning outputs are set to 3.3V and inputs are 3.3V-tolerant.
- **Outputs**
 - A GPIO pin designated as an output pin can be set to high (3.3V) or low (0V).
- **Inputs**
 - A GPIO pin designated as an input pin can be read as high (3.3V) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.
- **PWM (pulse-width modulation)**
 - Software PWM available on all pins
 - Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
- **SPI**
 - SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
 - SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
- **IIC**
 - Data: (GPIO2); Clock (GPIO3)
 - EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)
- **Serial**
 - TX (GPIO14);RX (GPIO15)
- **Use python to control IO**

```
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD
import time

mc = MyCobot(PI_PORT, PI_BAUD)

# Init
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
# HIGH
GPIO.output(20, 0)
GPIO.output(21, 0)
# WAIT
time.sleep(2)
# LOW
GPIO.output(20, 1)
GPIO.output(21, 1)
```

- **GPIO interfaces are defined in the following table:**

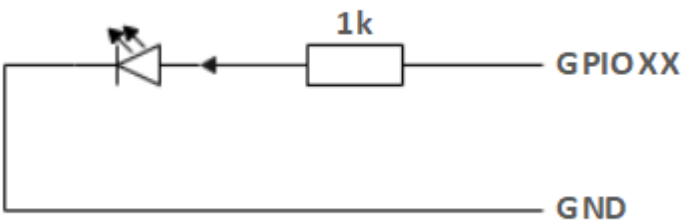
Table 3.5.2.1-1

Label	Signal	Type	Function	Remark
5V	5V	P	DC 5V	
5V	5V	P	DC 5V	
GND	GND	p	GND	
NC	NC	-	-	Currently not supported
NC	NC	-	-	Currently not supported
18	GPIO18	I/O	GPIO18	
GND	GND	p	GND	
23	GPIO23	I/O	GPIO23	
24	GPIO24	I/O	GPIO24	
GND	GND	p	GND	
25	GPIO25	I/O	GPIO25	
08	GPIO8	I/O	GPIO8	
07	GPIO7	I/O	GPIO7	
01	GPIO1	I/O	GPIO1	
GND	GND	p	GND	
12	GPIO12	I/O	GPIO12	
GND	GND	p	GND	
16	GPIO16	I/O	GPIO16	
20	GPIO20	I/O	GPIO20	
21	GPIO21	I/O	GPIO21	
3.3	3.3V	P	DC 3.3V	
NC	NC	-	-	Currently not supported
03	GPIO3	I/O	GPIO3	
04	GPIO4	I/O	GPIO4	
GND	GND	p	GND	
17	GPIO17	I/O	GPIO17	
27	GPIO27	I/O	GPIO27	
22	GPIO22	I/O	GPIO22	
3.3	3.3V	P	DC 3.3V	

Label	Signal	Type	Function	Remark
10	GPIO10	I/O	GPIO10	
09	GPIO9	I/O	GPIO9	
11	GPIO11	I/O	GPIO11	
GND	GND	p	GND	
00	GPIO0	I/O	GPIO0	
05	GPIO5	I/O	GPIO5	
06	GPIO6	I/O	GPIO6	
13	GPIO13	I/O	GPIO13	
19	GPIO19	I/O	GPIO19	
26	GPIO26	I/O	GPIO26	
GND	GND	p	GND	

Notice:

- 1. I/O: This function signal includes input and output combination.
- 2. When the single tube corner is set as the output terminal, it will output 3.3V voltage.
- 3. The source current of a single tube angle decreases with the increase of the number of pins, from about 40mA to 29mA.
- 4. If a certain GPIO is set to the output mode and outputs a high level signal, the circuit connected to the LED is shown in Figure and the LED will light up.



- 1. In the case of using other functions, the IO function is unavailable, and the other function table of the function interface is shown in Figure

3v3 Power	1	2	5v Power
GPIO 2 (I2C1 SDA)	3	4	5v Power
GPIO 3 (I2C1 SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (UART TX)
Ground	9	10	GPIO 15 (UART RX)
GPIO 17	11	12	GPIO 18 (PCM CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3v3 Power	17	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	20	Ground
GPIO 9 (SPI0 MISO)	21	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM DIN)
Ground	39	40	GPIO 21 (PCM DOUT)

3.2 Pedestal Introduction

- A. Figure 2.1.5.2-1 shows the front ports and buttons of the pedestal:

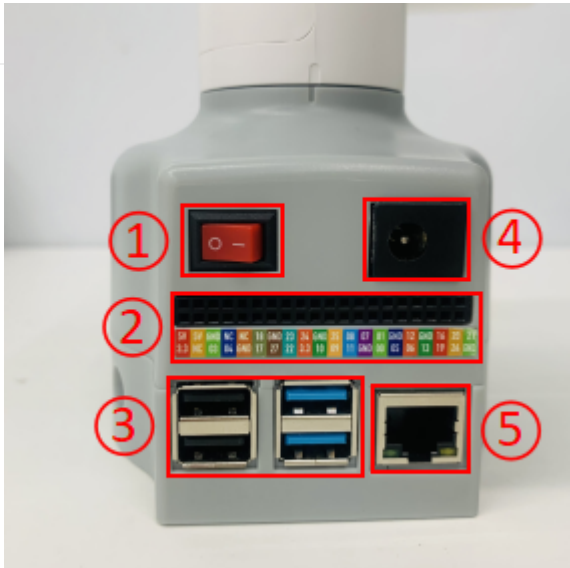


Figure 2.1.5.2-1 Front view of the base

- ① Switch
 - ② Functional Interface Group 1
 - ③ USB2.0 , USB3.0
 - ④ DC Interface of power
 - ⑤ The network interface
- B. Figure 2.1.5.2-1 Showing the ports on the side of the base:



Figure 2.1.5.2-2 side of base

- ① SD card slot
- ② Type C interface
- ③ HDMI interface
- ④ Headphone jack

3.3 Introduction to Bottom Electrical Interfaces

- A. Power DC interface: The myCobot280 is powered by a 6.5mm OD, 2.0mm OD, and a manufacturer's 12V 5A DC power adapter.

- B. Switch: Red means switch,also means powering on, while O means powering off.
- C. USB2.0 interface. Serial port with the standard of main line for 2.0 interface. The USB port is used to copy program files and connect peripherals such as mouse and keyboard.
- D. USB3.0 port (blue) : The port that uses serial bus 3.0 for data connection. The USB port is used to copy program files and connect peripherals such as mouse and keyboard.

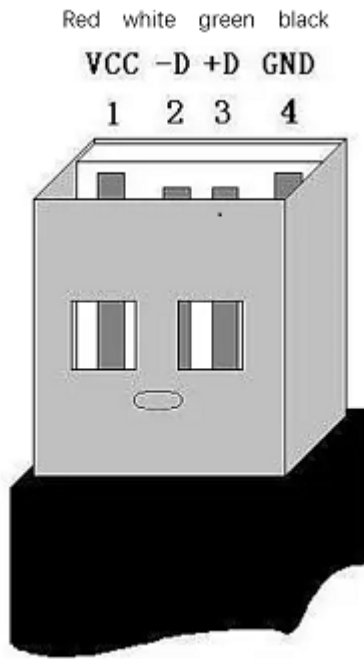


Figure 2.1.5.2-5

- E. The network interface: Ports for network data connection, as shown in figure 2.1.5.2-5. Ethernet interfaces can be used for communication between a PC and a robot system or for Ethernet communication with other devices.

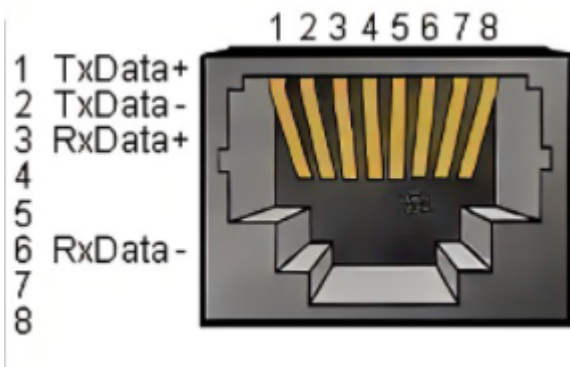


Figure 2.1.5.2-6

- F. HDMI port: The HDMI D-type port connects with the monitor. HDMI port 2 has a priority, and HDMI port is recommended.
- G. Type-C port: connecting to the PC.
- H. SD card slot: The SD card can be inserted and removed. The size of the SD card is 32mm x 24mm x 2.1mm

Downloads

Brochure Download

Robot	Brochure
myCobot 280 M5	download
myCobot 280 PI	download
myCobot 280 AR	download
myCobot 280 JN	download
myPalletizer 260 M5	download
myPalletizer 260 PI	download
mechArm 270 M5	download
mechArm 270 PI	download
myCobot 320 M5	download
myCobot 320 PI	download
myCobot Pro 600	download
myAGV	download
marsCat	download
metaCat	download
人工智能套装 2023	download
myarm 300PI	download

3D model files

3D model files of robots

Robot	3D model file
myPalletizer 260	download
mechArm 270	download
myCobot 280 M5	download
myCobot 280 PI	download
myCobot 280 Arduino	download
myCobot 280 JN 2023 ver.	download
myArm 300	download
myCobot 320 M5 2020 ver.	download
myCobot 320 M5 2022 ver.	download
myCobot 320 M5 2022 ver. v1.2	download
myCobot 320 PI 2022 ver.	download
myCobot 320 PI 2022 ver. v1.2	download
myCobot Pro 600	download
myBuddy 280	download
myAGV	download
myAGV 2023	download

Robot 2D drawing

Robot	Robot 2D drawing
myPalletizer 260 Atom	download
myPalletizer 260 PI base	download
myPalletizer 260 M5 base	download
mechArm 270 Atom	download
mechArm 270 PI base	download
mechArm 270 M5 base	download
myCobot 280 Atom	download
myCobot 280 X3pi base	download
myCobot 280 M5 base	download
myCobot 280 JN base	download
myArm 300 Atom	download
myArm 300 base	download
myArm 300 joint	download
myCobot 320 Atom	download
myCobot 320 PI base	download
myCobot 320 M5 base	download
myCobot Pro 600 Atom	download
myCobot Pro 600 base	download
myAGV 2023	download

3D models of accessories

- myCobot series

Accessories	3D model file
Adaptive gripper	download
Parallel gripper	download
Flexible gripper	download
G-base 2.0	download
Large suction cup base	download
Flat base	download
Vertical suction pump	download
Double suction pump	download
Camera flange	download
Asparagus flange	download
Dexterous hands	download
Pen holder	download
Phone holder	download

- myCobot Pro series

Accessories	3D model file
Electric gripper	download
Pneumatic gripper	download
Flexible gripper	download
Module suction cup	download
Adaptive gripper	download

Image Download

Product	Syetem Version	Link	SHA256 Hash
AI Kit 280	ubuntu 18.04	download	d44439be351a52decdb4470cb623a032047e223ffce73477d29aa9
myCobot 280 PI	ubuntu 18.04	download	04e40af5b637ec003a8b23ef9012e353361fd336db4e17cf9a65feb
	ubuntu 20.04	download	ce666e6c1047c512fe6b270336d472e48f231be12808729ed57f74
myCobot 280 JetsonNano	ubuntu 18.04	download	2f1e40c1480b077bcc83abd3b79ac175f25d21e9cc344a01463616
	ubuntu 20.04	download	8c6f03d2439a2dbdaf1dd689da903ba6b8c5d0665c395954d27bct
myCobot 320 PI	ubuntu 18.04	download	bc2ed6ef8d51a885f45379392b71e35420638a427d5b4b3a3c9d1f
	ubuntu 20.04	download	c95633bfd49246254f2be4783c6a91a15212422219157962c9312f
myPalletizer 260	ubuntu 18.04	download	f6fe999519146428e4c60960b242f647ae5c73c704852d686b2858
	ubuntu 20.04	download	04ebbad80926bc7b5d94a4ecb6e64a2205babb1564af07dd45266
mechArm 270	ubuntu 18.04	download	9af1fcbf9c608eda269dc395a8d68ea0a270008a88ec8ec3cf97758
	ubuntu 20.04	download	cbdc05cee595e9955f2d2c8c4a8ed13cdb95db125f6be1be6316bd
myCobot Pro 600	Raspberry Debian	download	2e73aaa153bddbf0a49d18669a254b27403f17f8e989c05d13836d
myArm 300	ubuntu 20.04	download	1325433fec7ec7d028f468315a832210fd9ce004a6dd07eb859afc7
myBuddy 280	ubuntu 20.04	download	2b5452f665bcb999faf1727b2103dc1e5745705f5706728e140d62f
myAGV	ubuntu 18.04	download	bedad7d9769cb69380c6a4b9742ba7aefc21db41ab239172b7a5a

5 Development and Use based on Drag-and-drop Programming Software



What's drag-and-drop programming?

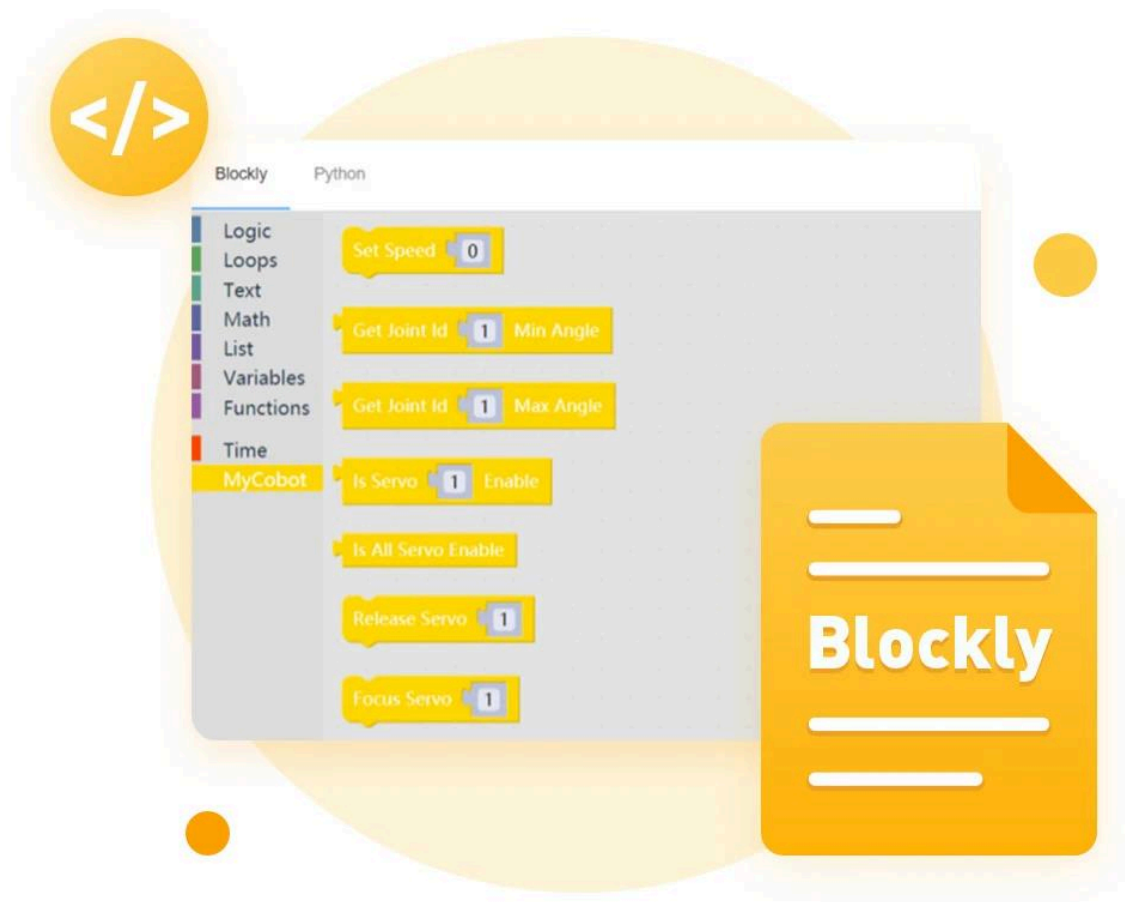
"Drag-and-drop" programming is a basic technology which allows programming by dragging and dropping code building blocks or using other vision clues instead of writing text-based codes manually. Therefore, it makes complex and abstract programming languages easy to understand.

Hadi Partovi, the founder of Code.org, once said "that drag-and-drop programming allows you to understand programming basics easily without carefully studying character arrangement. Once knowing the basic concepts to use drag and drop, you will immediately want to learn how to use them to do actual things." This completely follows the basic logic for learning programming.

A number of programs can help you get started with drag and drop programming, which include Google Blockly, Microsoft MakeCode, MIT Scratch, etc.

What are myBlockly and UIFlow respectively

Both **myBlockly** and **UIFlow** implement the model of drag-and-drop programming. The former is developed by the R&D team of Shenzhen Elephant Robotics Company, and the latter is a graphical programming platform developed by M5Stack.



myBlockly is a piece of puzzle programming software based on python environments and pymycobot dependent libraries, which allows the user to do programming in a block building way to control mycobot. It is suitable for programming learners and can train their programming logic skills. myblockly also provides a python display interface, which can convert the myblockly program built by the user to a python language one to help the user learn python statements.



However **UIFlow** is a programming tool specially designed for M5 hardware system, which uses Blockly graphical programming and Python code programming. It not only provides programming learners with friendly programming experience, but also delivers more professional and crucial programming knowledge.

First-Time Use

Preparing for the first-time use

Follow the steps below to download Python and burn firmware via MyStudio.

Step 1 Download Python. myBlockly is used in Python environment. Make sure Python is downloaded on PC ([Reference for information about configuration of Python environment](#)).

Step 2 Burning firmware.

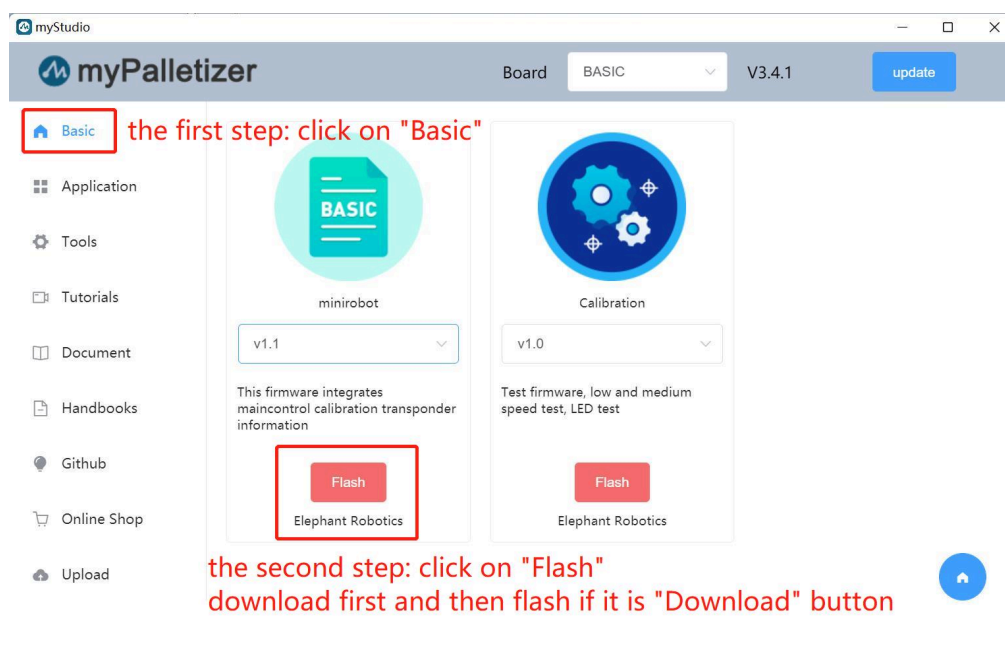
- Before burning firmware, download corresponding serial port driver according to the USB chip on your PC. CP210X is suitable for CP2104 version and CP34X is suitable for CH9102 version. You can install both of them if you are unable to confirm the type of USB chip. Go to the address below to download and install serial port driver.
 - serial port driver for M5Stack-basic:
 - CP210X: [Windows 10](#), [MacOS](#), [Linux](#)
 - CP34X: [Windows 10](#), [MacOS](#)
 - serial port driver for Atom:
 - [Windows 10](#)

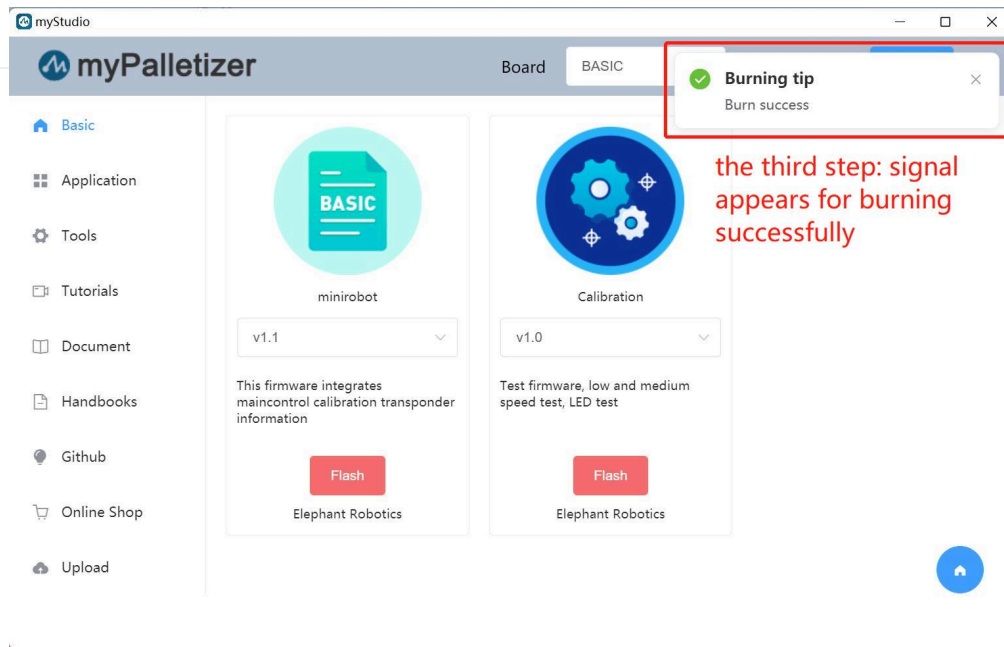
Notice:

- An error may be reported during installation of CH9102_VCP_SER_MacOS. However, the installation already completes. Just ignore the error.
- Go to [4.1.1 MyStudio Installing driver](#) for specific information about serial port driver installation.
- Make sure that settings of the system "\"Preferred settings -> Security and privacy -> General\" is correct and allow the user to get it from App Store or an approved developer.

Step 3 M5Stack-basic burning is not required on Raspberry Pi series and JETSON NANO series, and their lasted atomMain on Atom is factory burnt.

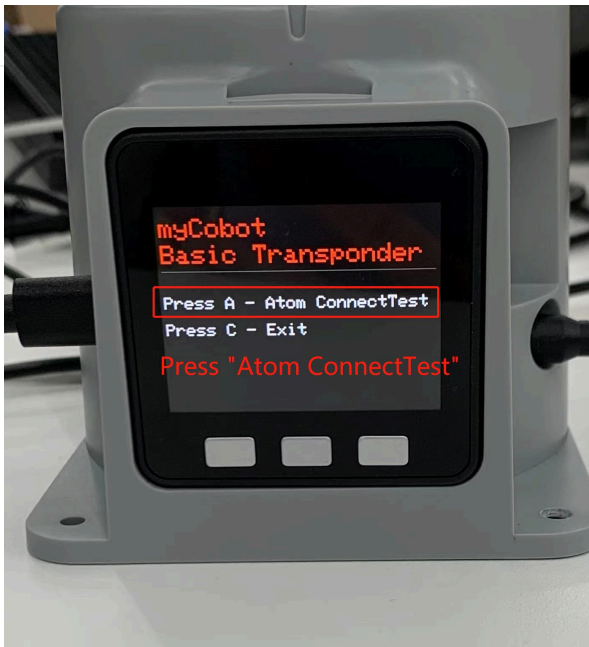
Step 4 For M5Stack series, miniRobot is required to be burnt on M5Stack-basic via MyStudio. The lasted atomMain on Atom is factory burnt. Go to [4.1 MyStudio](#) for its download and installation. Follow the steps below to burn miniRobot.





Step 5 Press "Transponder" button on M5Stack-basic, and then press "Atom ConnectTest".





Installing myBlockly

Go to the addresses below to download myBlockly.

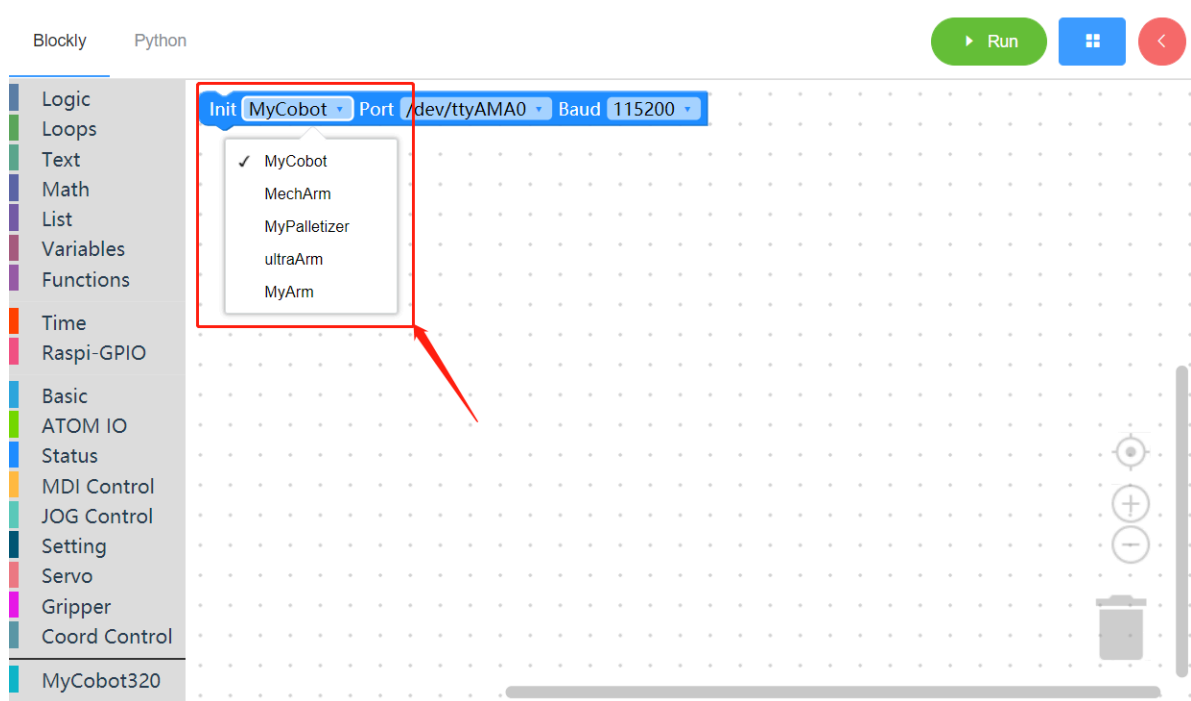
- [Github](#)
- [Official Web](#)

Notice:

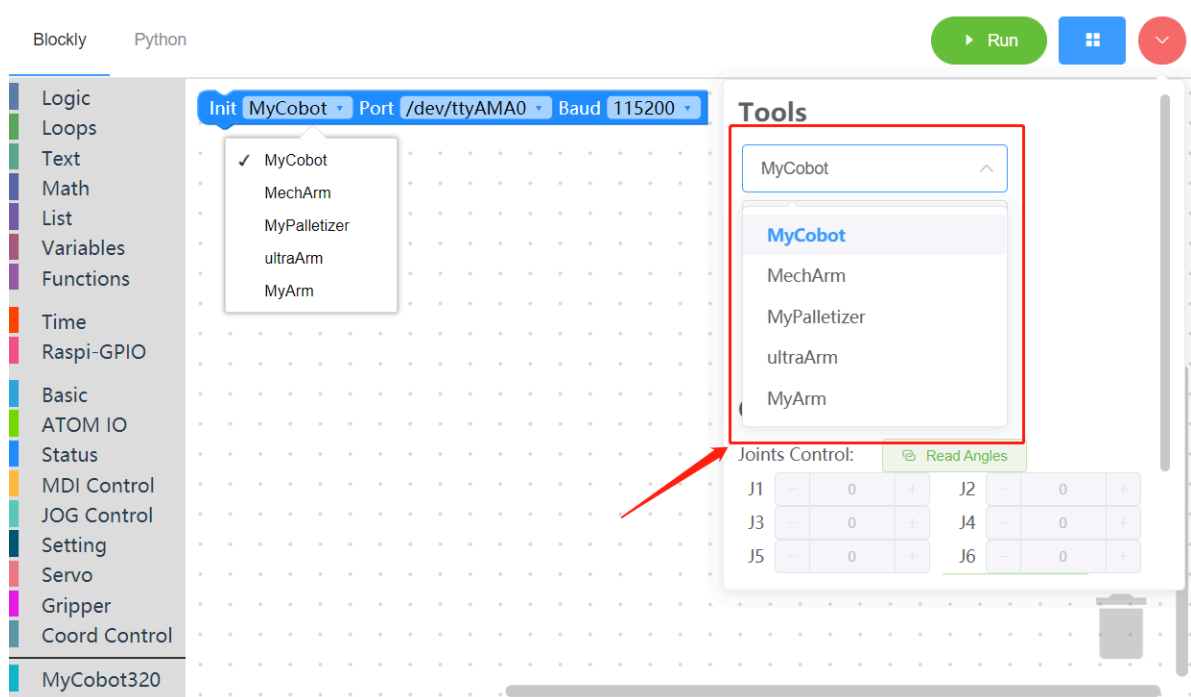
Make sure to download the latest version.

Before use

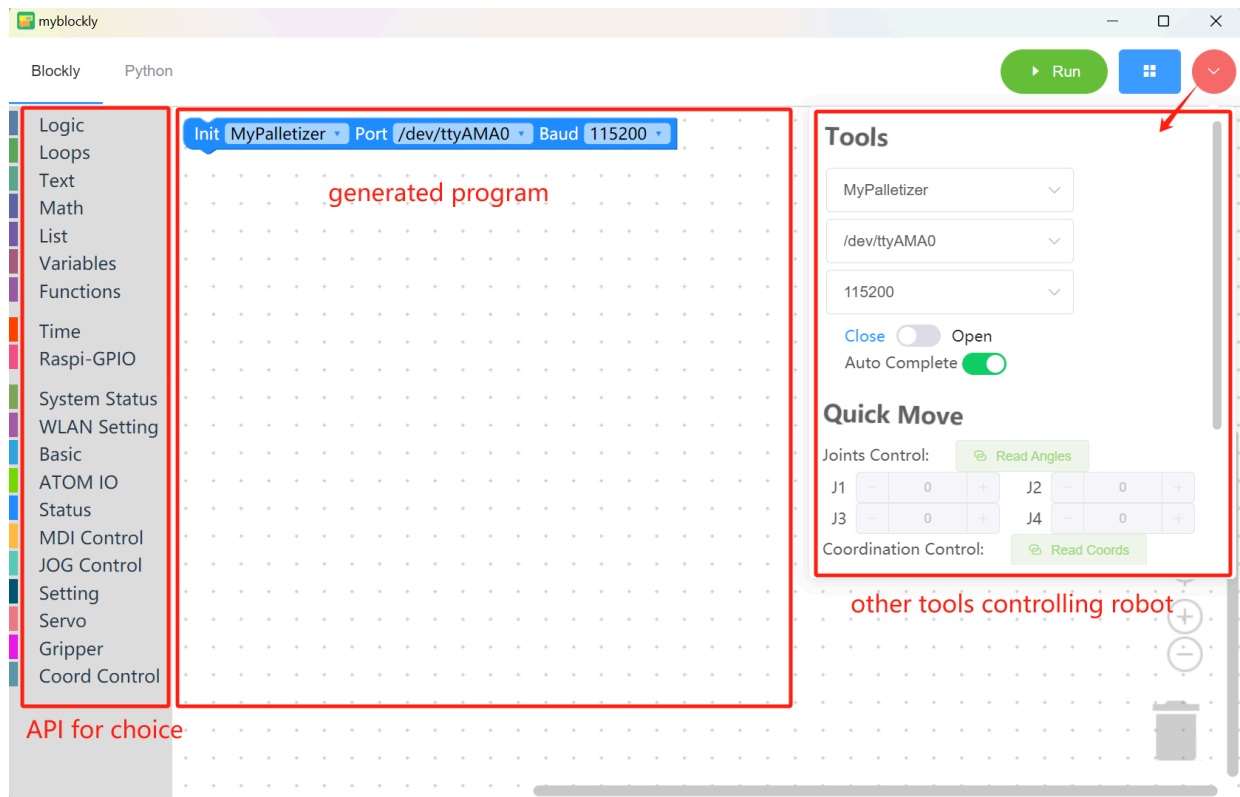
- Before start to programming, be sure to select the **corresponding machine model**, otherwise it is easy to cause hardware damage



- When controlling robot by quick move, be sure to select the **corresponding machine model**, otherwise it is easy to cause hardware damage



myBlockly View



- **API blocks:**

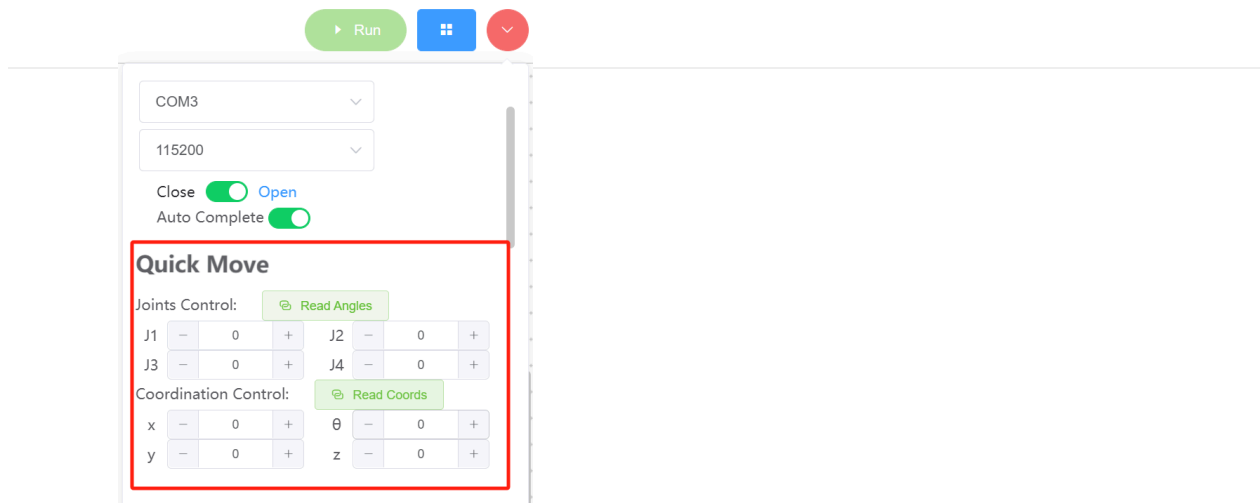
- Display API used to generate programs
- Contains method modules required for programming, which can be concatenated by placing the mouse into the program editing area

- **Generated program:**

- Before running the program, it is necessary to select the correct model, port, and baud rate in the initialization module or other tools controlling, otherwise the program cannot run normally.
- Drag and drop the required module methods to the area and concatenate them to implement your own program.
- If the baud rate and serial port have been modified in the right toolbar, but the current address is still /dev/ttyAMA0, it is due to the myBlockly version issue. You need to update the software version on the official website first (the latest version will change the information in the editing area after selecting the serial port and baud rate in the toolbar).

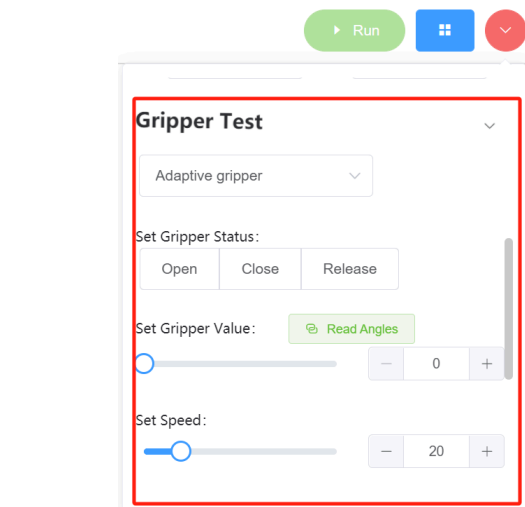
- **Other tools controlling robots:**

- Click on the pink icon in the top right corner to select the corresponding robot type, serial port and baud rate.
- Other tools — Quickly move



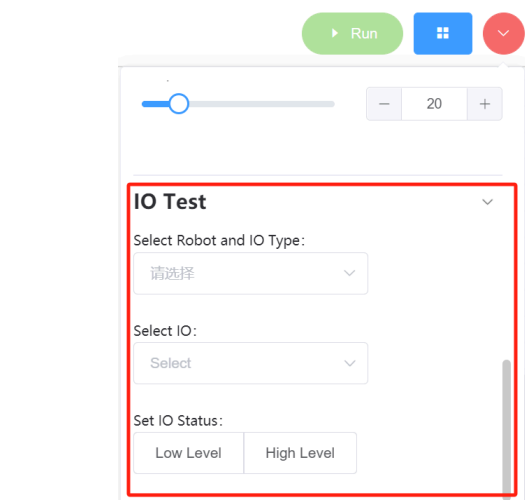
Find the Quick Move module below in the other toolbars. Tap **Joints Control** OR **Coordination Control** for joints' real-time angles and coords. Tap **+** or **-** to control robot movements.

- Other tools — Gripper Test (At present, the gripper test function only adapts to the following models: MyCobot 280 series, MeCharm 270 series, MyPalletizer 260 series, MyARM 300 PI.)



Find the Gripper Test module in other toolbars. You can set the type of gripper and the movement speed of the gripper. Click the 'Read Angle' button to obtain the current value of the gripper. Set gripper status: open, close, release. Pay attention to the movement of the gripper. (Please refer to the specific testing steps--[Gripper Test](#))

- Other tools— IO Test (At present, the IO test function only adapts to the following models: MyCobot 280 series, MeCharm 270 series, MyPalletizer 260 series, MyARM 300 PI.)



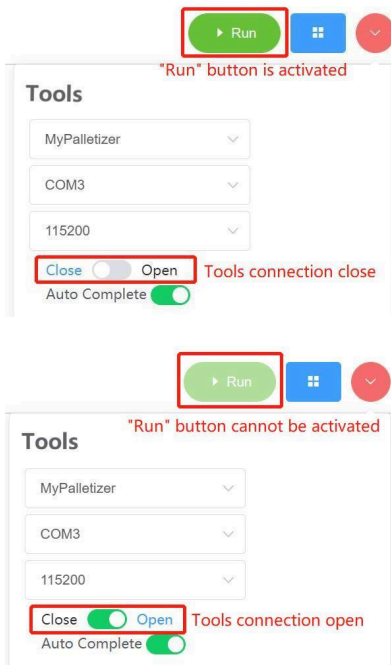
Find the IO test module in other toolbars. Select the machine and IO type, and choose the IO value. You can click the low or high button to change the current selected IO port status. (Please refer to the specific testing steps--IO test)

Notice:

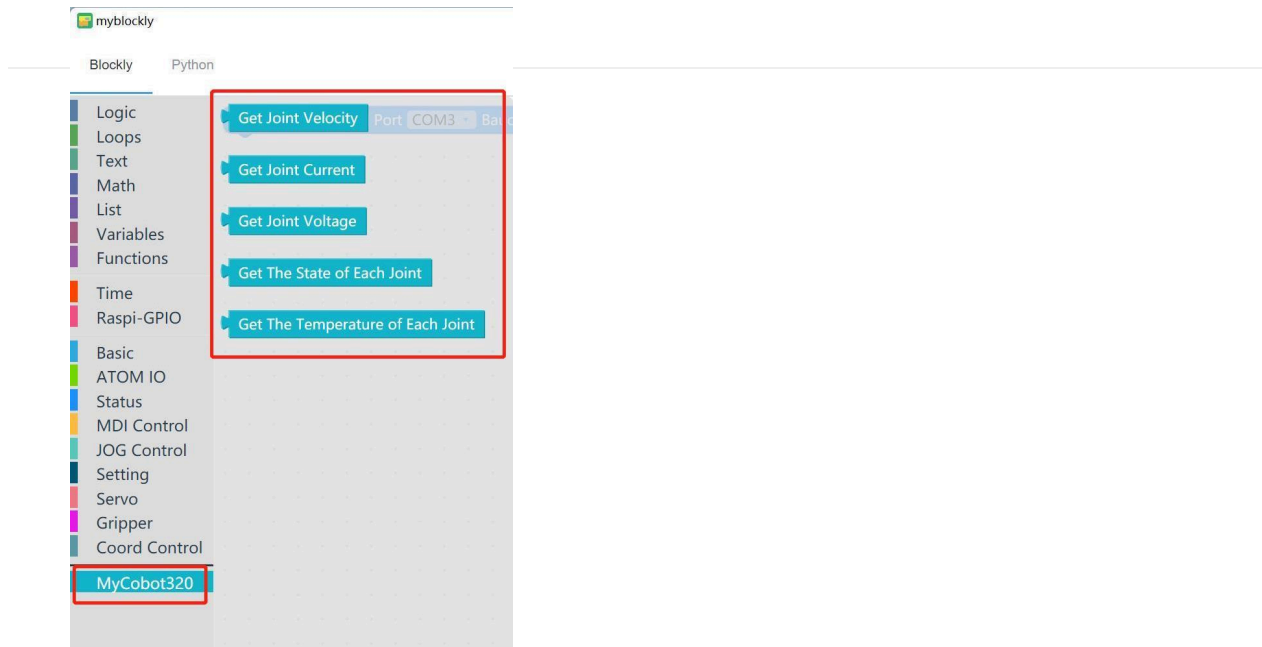
1. Baud rate for M5Stack series and Raspberry Pi series is 115200 and 1000000 respectively.

Robot	Port	Baudrate
260 M5	Win: COM; Linux: /dev/ttyUSB;	115200
270 M5	Win: COM; Linux: /dev/ttyUSB;	115200
280 M5	Win: COM; Linux: /dev/ttyUSB;	115200
320 M5	Win: COM; Linux: /dev/ttyUSB;	115200
260 PI	/dev/ttyAMA0	1000000
270 PI	/dev/ttyAMA0	1000000
280 PI	/dev/ttyAMA0	1000000
320 PI	/dev/ttyAMA0	115200
280 Jetson Nano	/dev/ttyTHS1	1000000
280 Arduino	Win: COM; Linux: /dev/ttyUSB or /dev/ttyACM* ;	1000000

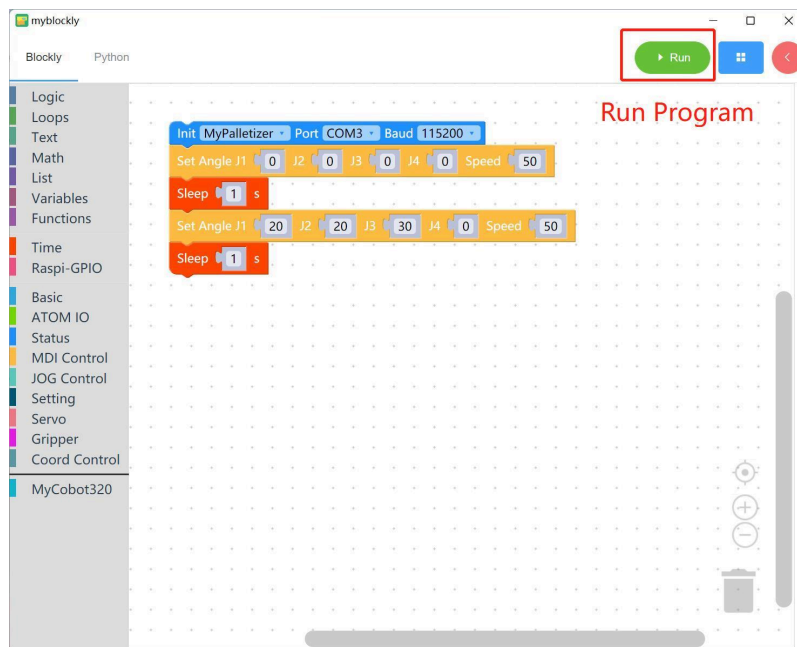
1. Go to [4.1.1 Download and Installation of myStudio](#) for serial port and baud rate of different types of robots.
2. Check connection of **Tools** in top right corner if program cannot run successfully.



3. API for MyCobot 320 is specifically set in the bottom left corner. It does not affect any program.



Running Program

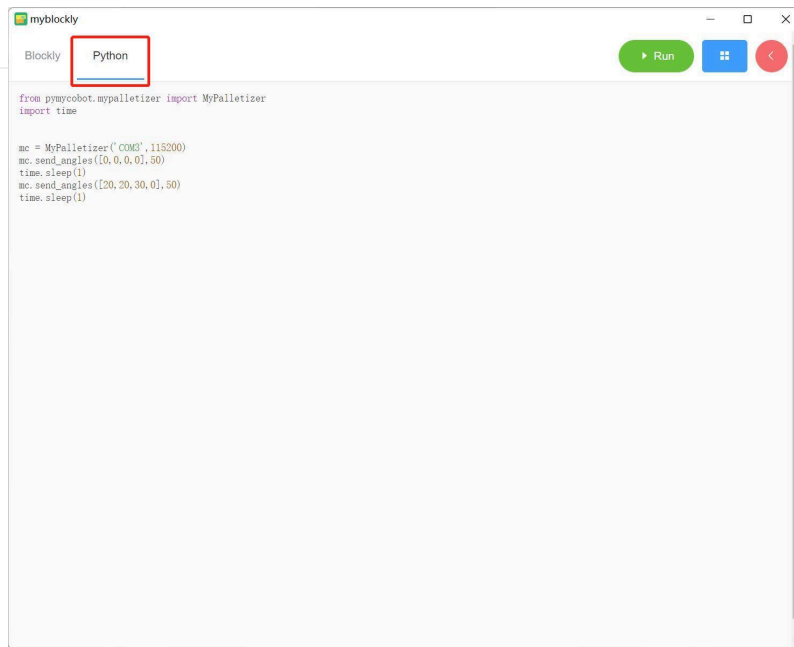


Drag API blocks from the left side to integrate them with each other for program edition. Then, tap "Run" button to run the program.

Notice:

`Sleep` block is required to provide no less than 0.5 second for a robot to move. Otherwise the robot cannot run successfully.

Click on `Python` for Python codes corresponding to blocks edited in myBlockly.

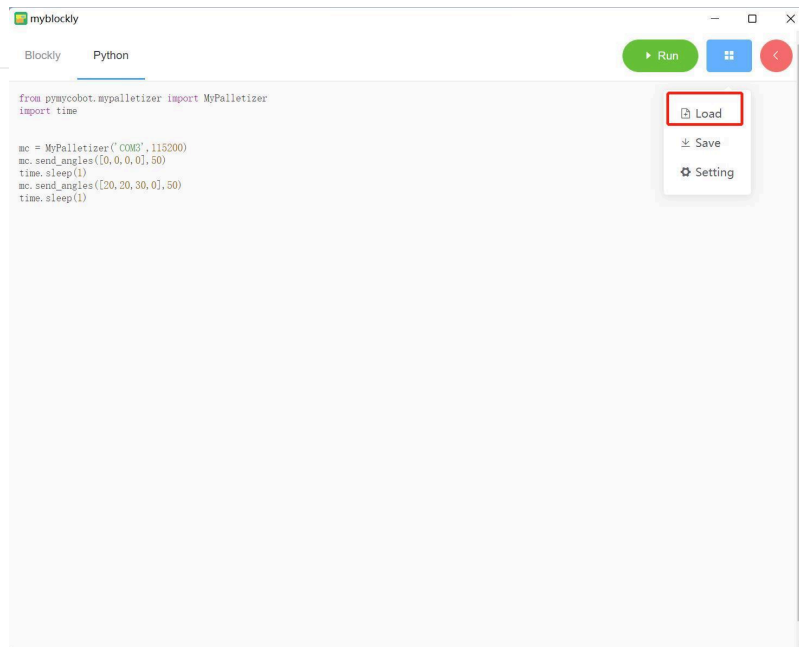


Saving and Loading Programs

- myBlockly files are saved as "*.json" format. Click on the blue button in the top left corner, and then click on "Save".



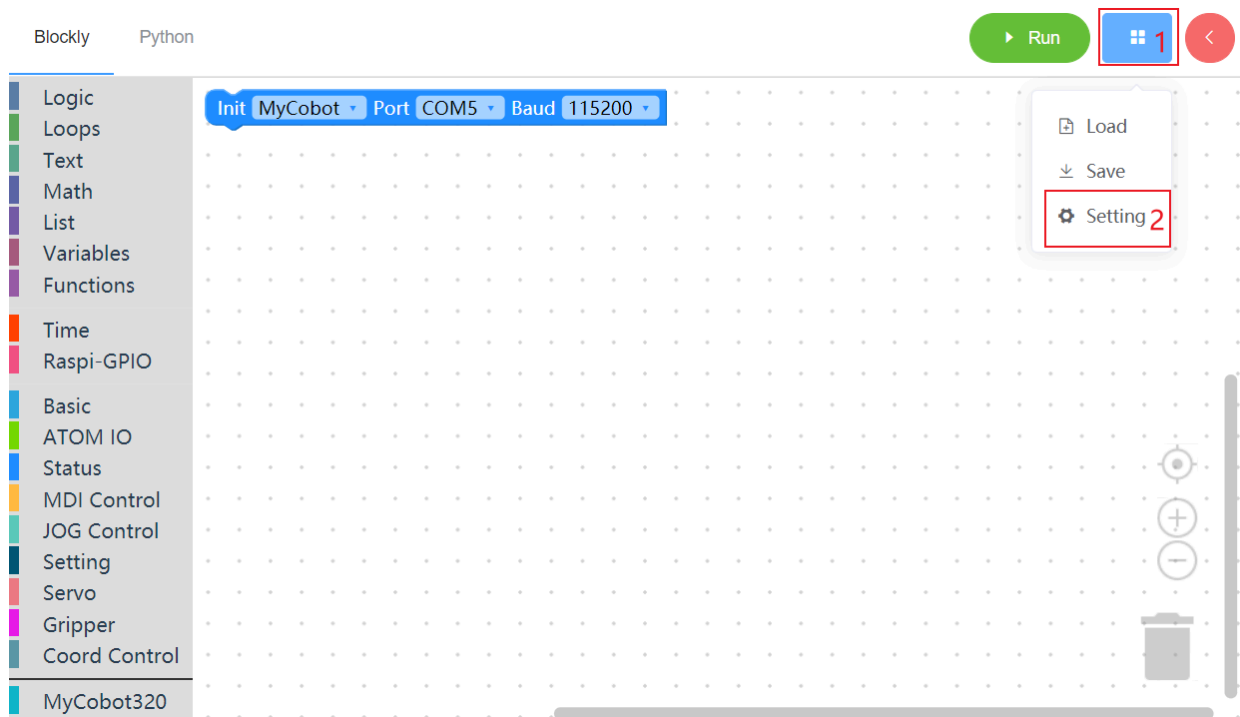
- Click on "Load" button to import existing programs.

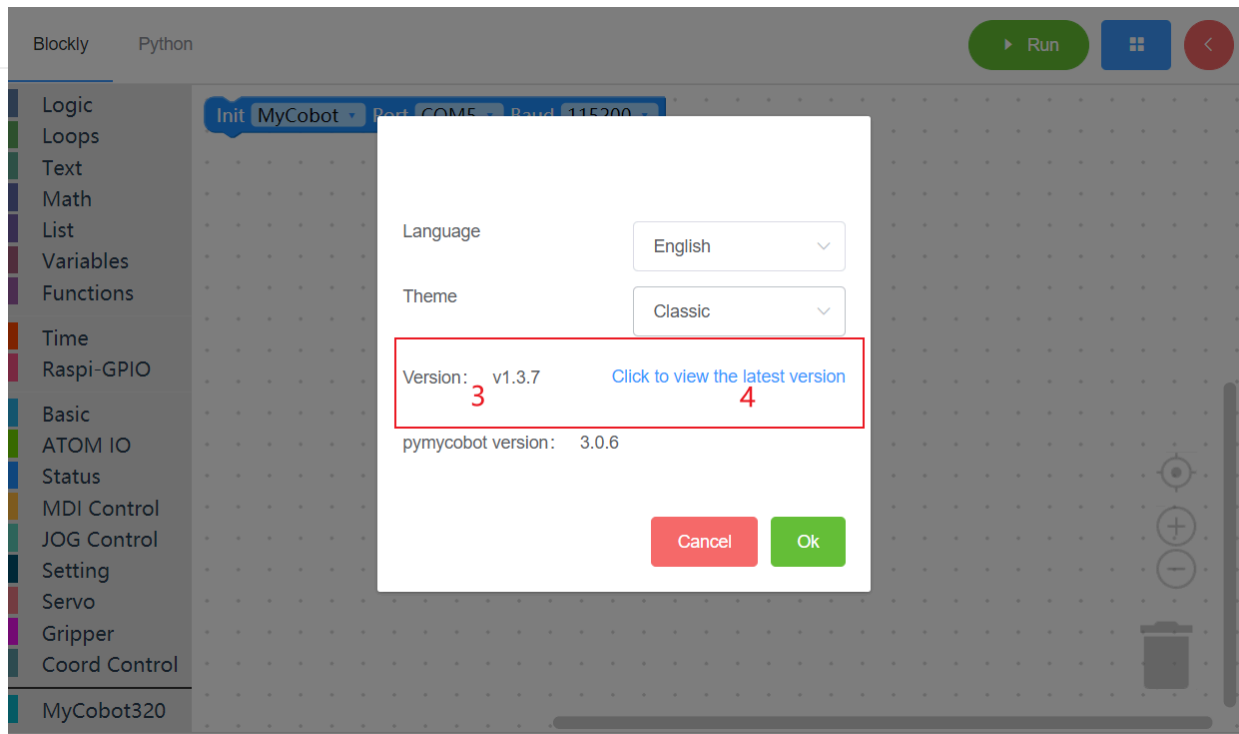


How to install and update the software

The installation and update of myBlockly need to go to the [official website](#) to download the latest version.

Inside the software, you can update the software through the following steps.





- 1 Mouse move here
- 2 Click setting
- 3 This is your current version
- 4 Click this link to view the latest version and download

And if you need to know the detailed update log of the software, you can go to [github](#) to view.

How to run the sample code

We provide a wealth of sample codes, please select the code you are interested in, then find the sample building blocks from the left menu bar, drag them to the workspace, and click Run to see the result.

- [Setting the Color of RGB Light Panel](#)
- [Setting All Arms to Starting Point](#)
- [Controlling Single-Joint Motion](#)
- [Controlling Multi-Joint Motion](#)
- [Swinging Arms Left and Right](#)
- [Let Robot Dance](#)
- [The Use of Gripper](#)
- [The Use of Sucking Pump](#)

How to view the drive library open interface

Myblockly implements all interfaces of [pymycobot](#). You only need to find the corresponding block of the interface you want to use in the menu bar on the left side of the workspace, and then drag it to the workspace to use it.

Setting the Color of RGB Light Panel

Preparation

M5Stack series: Make sure robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

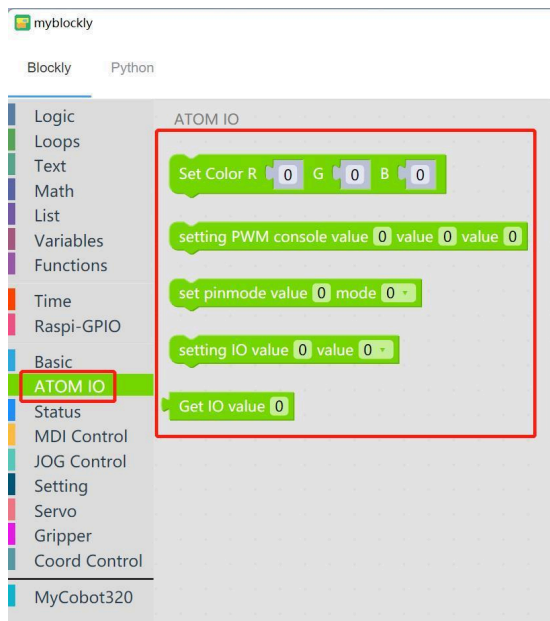
Other series: Make sure the robot is in normal status.

Purpose for this section

This section includes instructions for controlling the RGB light panel.

Introduction to API

- Set Color R() G() B()



- Applicable to myCobot 280 series, myCobot 320 series, myCobot Pro 600 series and myPalletizer 260 series



- Parameters:

Set the color of RGB light panel. R , G and B means red, green and blue respectively.

Brightness Range:

R: 0-255

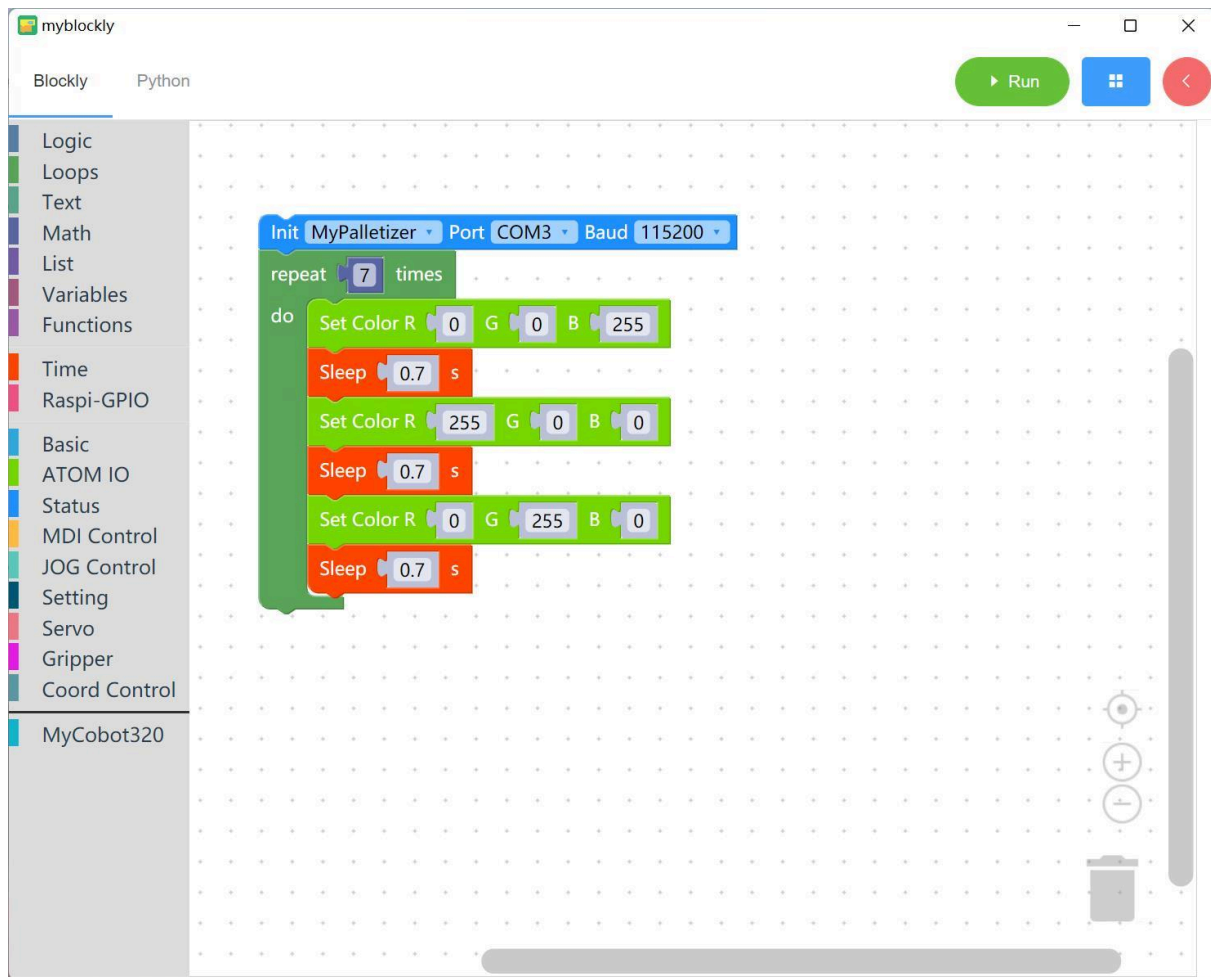
G: 0-255

B: 0-255

- Function: Controlling RGB light panel

Simple Demo

- program for display



- Change in color of RGB light panel:

The color of RGB light panel changes in sequence of blue, red, then green. The whole process loops 7 times.

Setting All Arms to Starting Point

Preparation

M5Stack series: Make sure robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

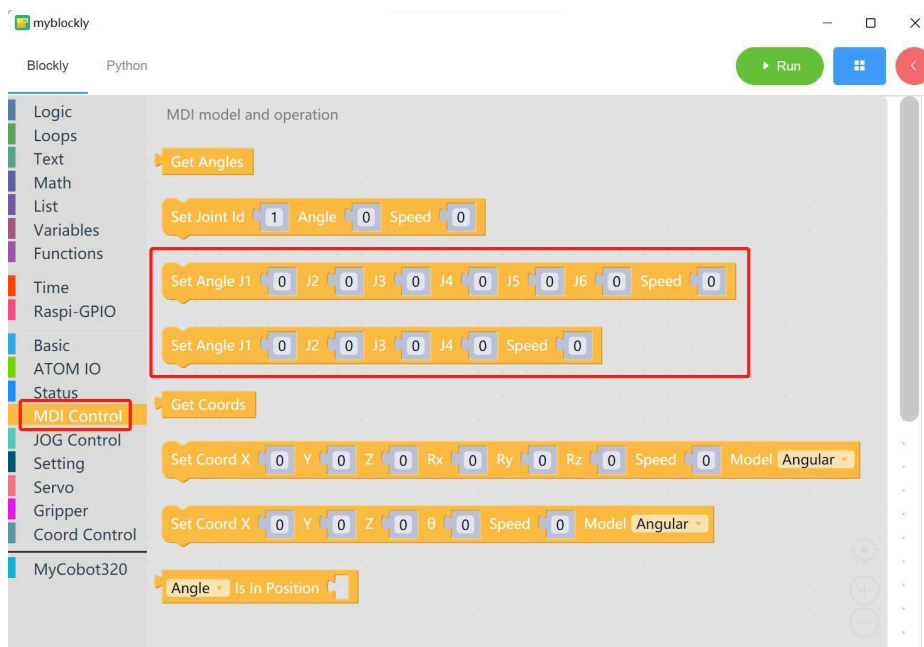
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for setting all arms to starting point.

Introduction to API

- Set Angle ()



- Applicable to six-axis robots (myCobot 280 series, mechArm series and myCobot 320 series)



- Applicable to four-axis robots (myPalletizer series)



- Parameters:

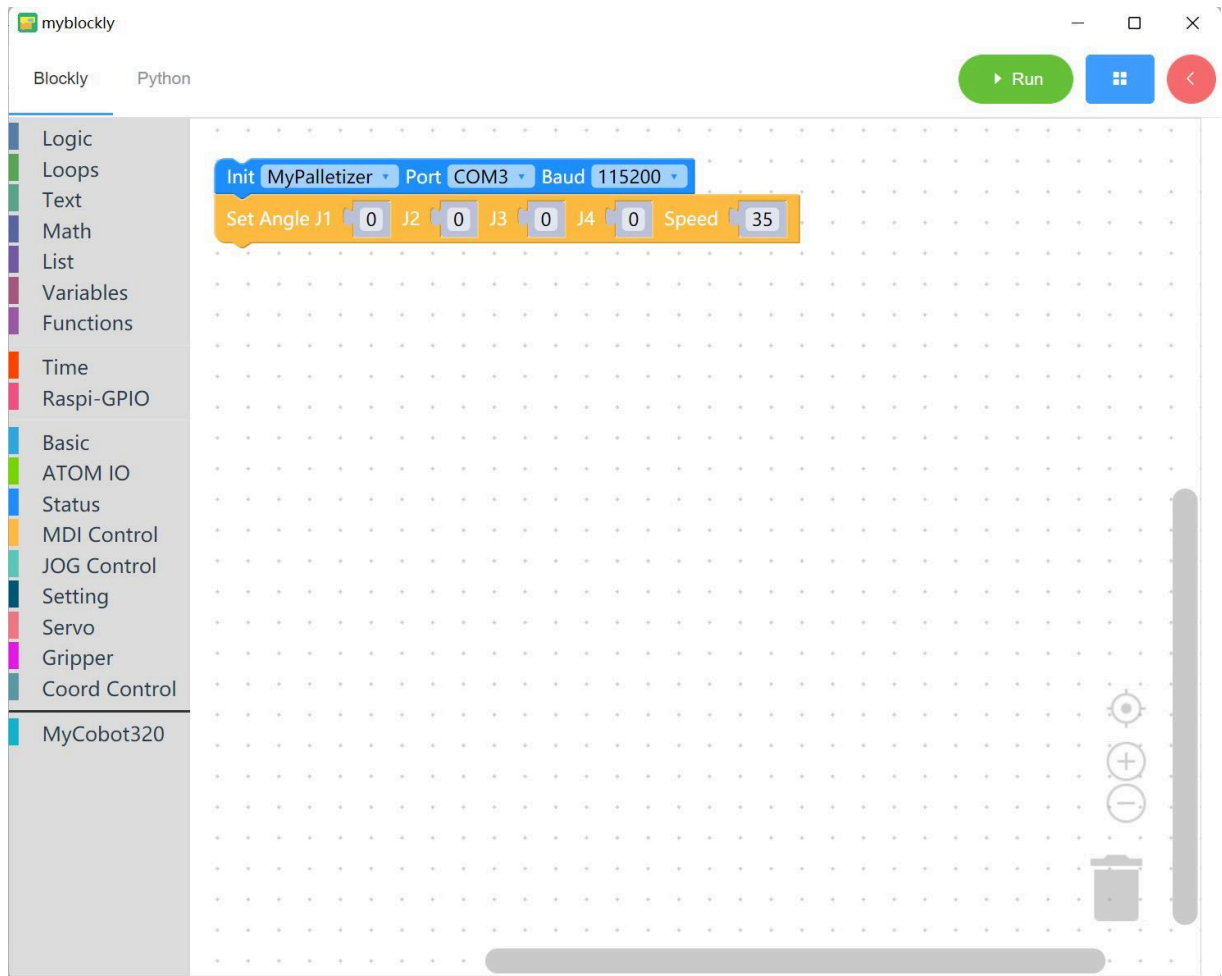
Joint angle: if arms are set to starting point, set parameters of angles as 0

Speed: Refer to [2 Products Profile](#)

- Function: Set all arms to the starting point (Angle 0)

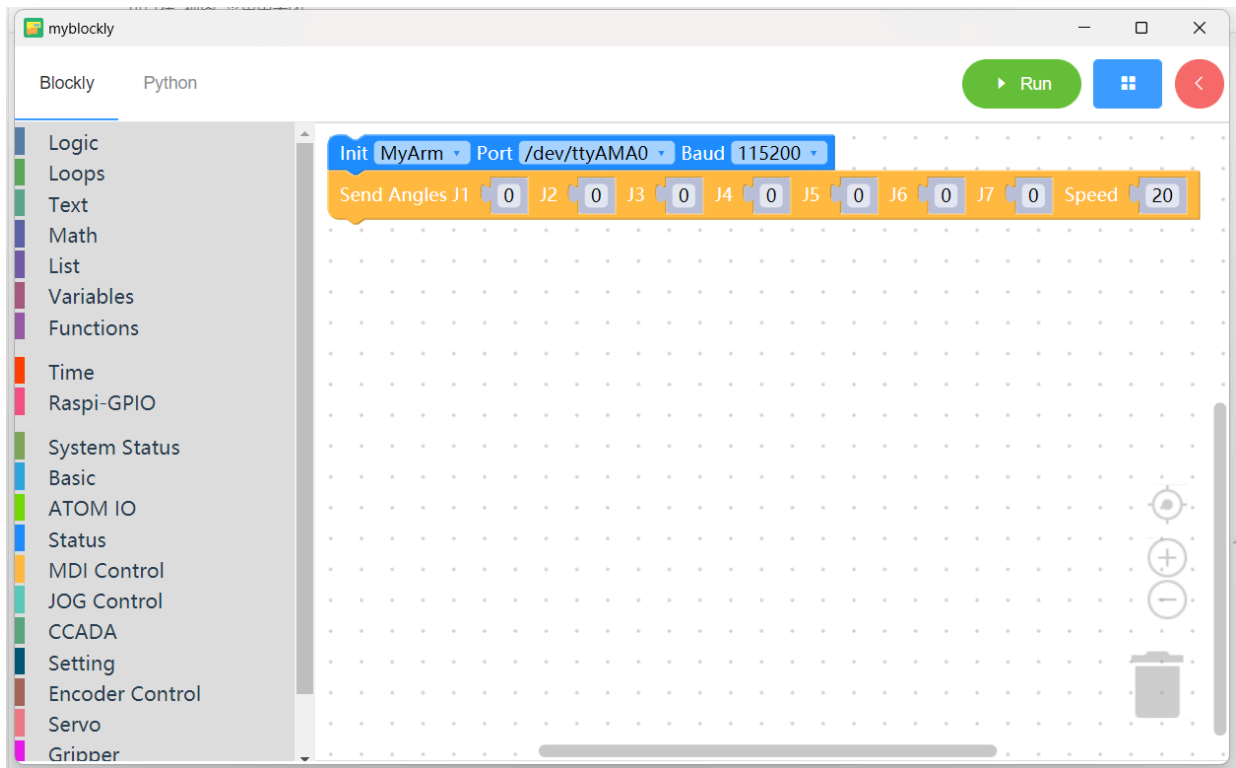
MyPalletizer

Simple demo



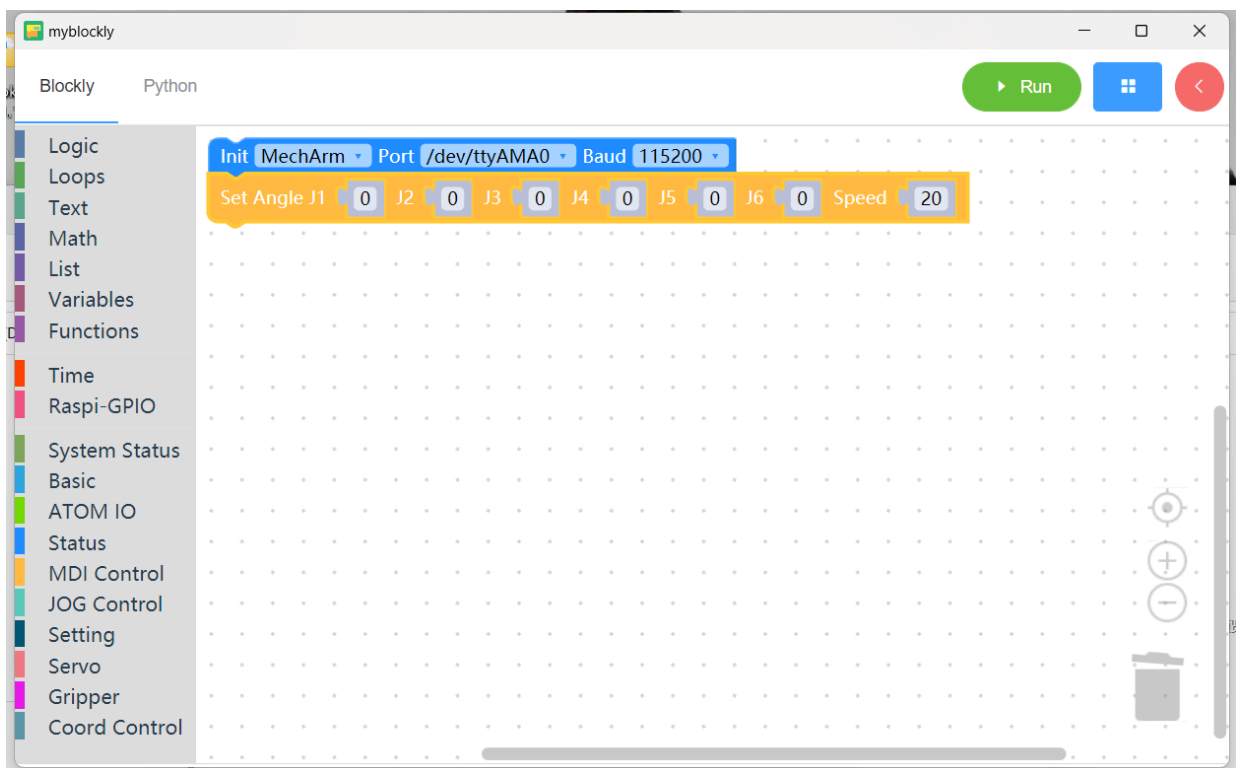
MyArm

Simple demo



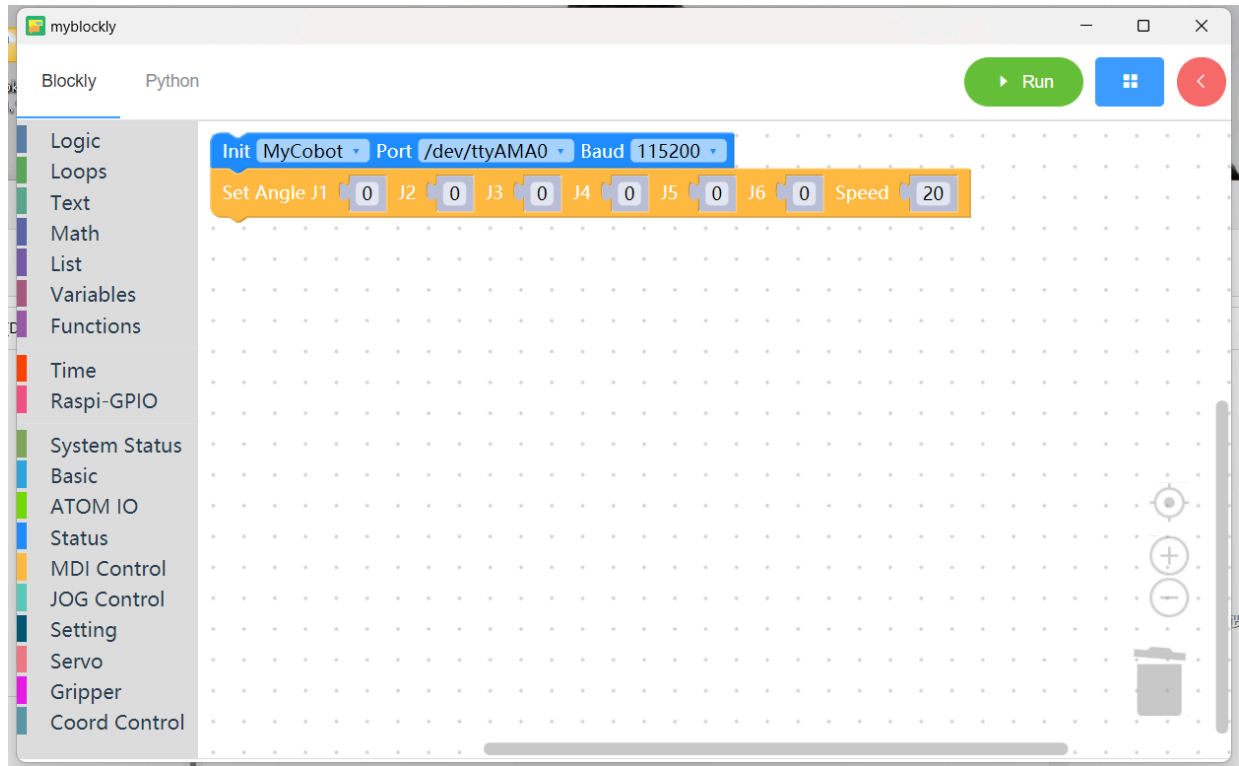
MechArm

Simple demo



MyCobot

Simple demo



- Motion:Six

All arms move to its starting point in 35 seconds.

Controlling Single-Joint Motion

Preparation

M5Stack series: Make sure robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

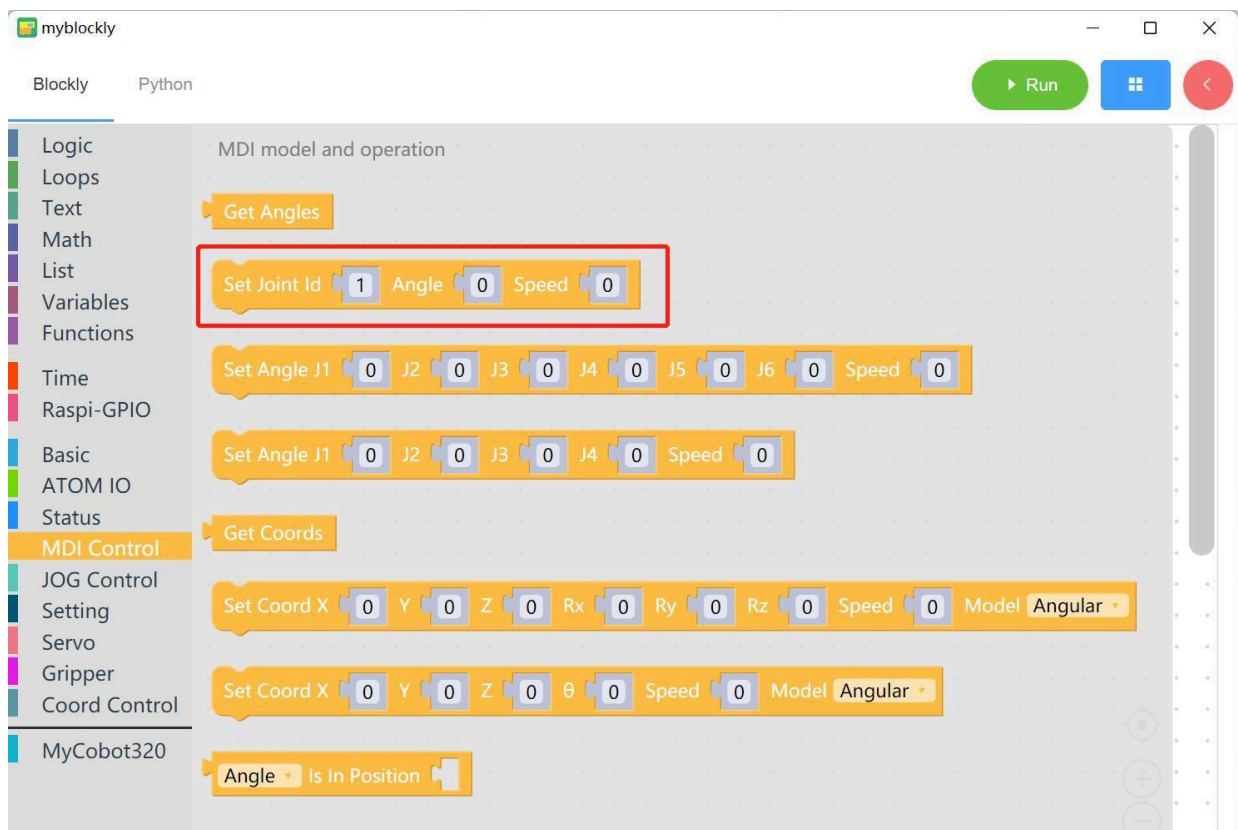
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for controlling single-joint motion.

Introduction to API

- Set Joint Id x Angle x Speed x



- Applicable to myCobot 280 series, myCobot 320 series, myCobot Pro 600 series and myPalletizer 260 series



- Parameters:

Joint Id range: 1-6 (myCobot 280 series, myCobot 320 series and myCobot Pro 600 series)

1-4 (myPalletizer 260 series)

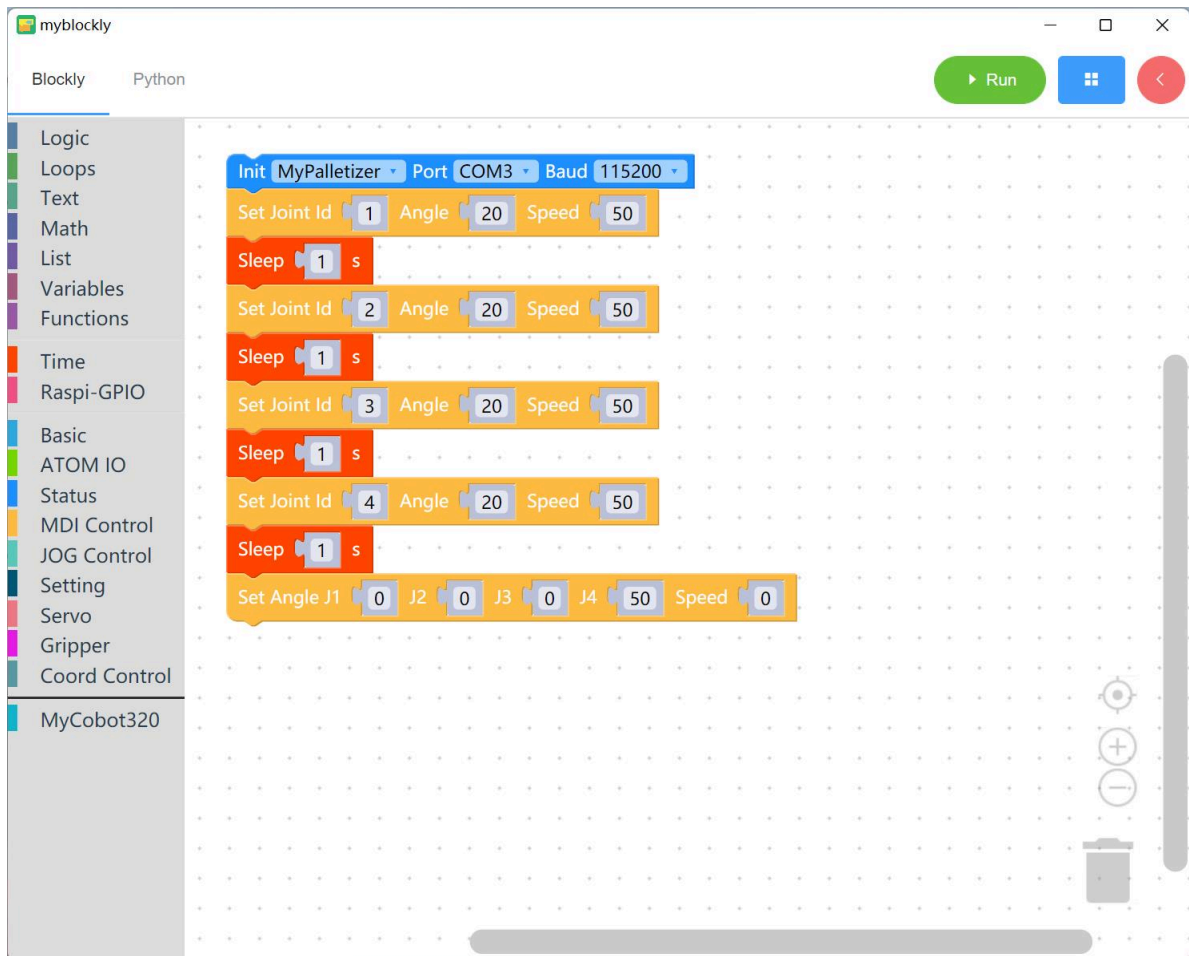
Angle: Refer to [2 Products Profile](#)

Speed range: 1-100

- Function: controlling single-joint motion

Simple Demo

- Program for display:



- Motion:

Joint 1 move to 20 degree angle at the speed of 50,

Joint 2 move to 20 degree angle at the speed of 50 in 1 second,

Joint 3 move to 20 degree angle at the speed of 50 in 1 second,

Joint 4 move to 20 degree angle at the speed of 50 in 1 second,

all arms move to starting point in 1 second.

Controlling Multi-Joint Motion

Preparation

M5Stack series: Make sure robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

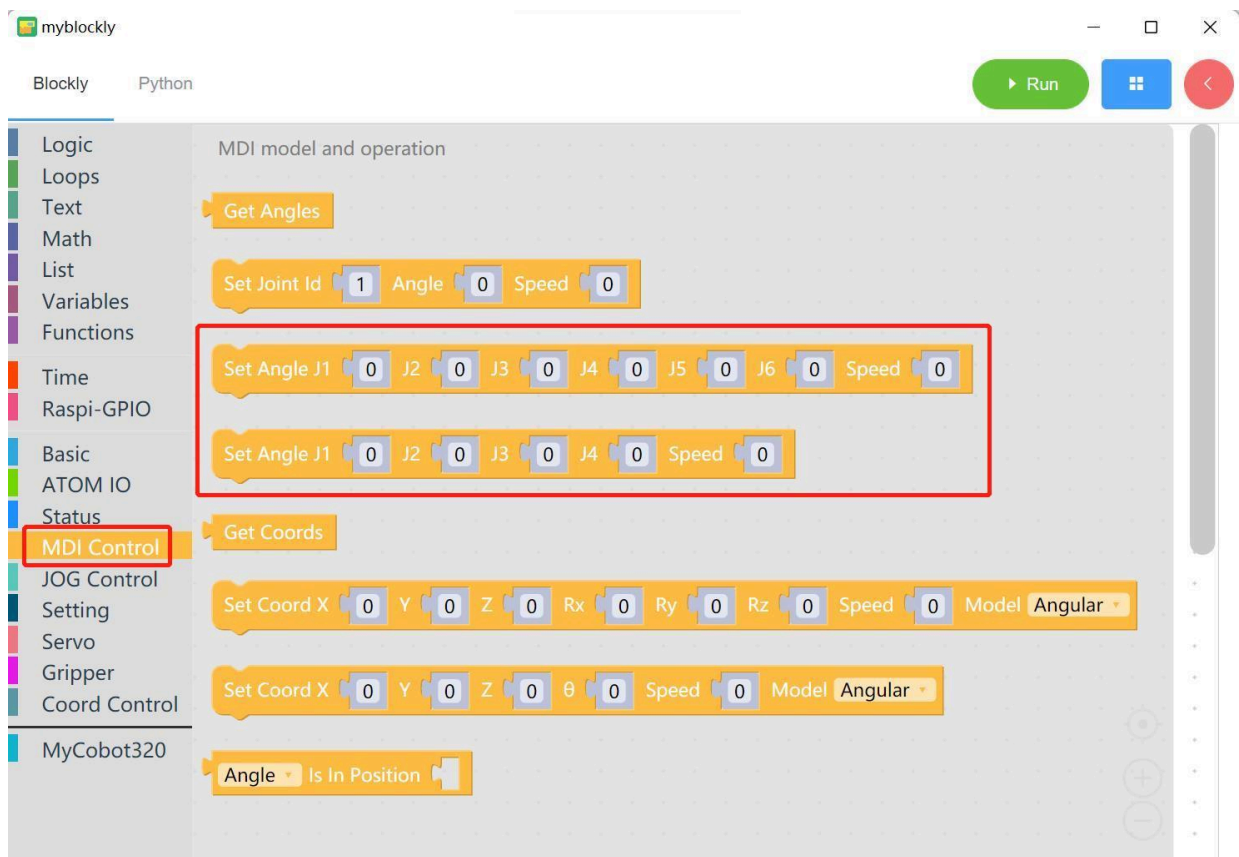
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for controlling multi-joint motion.

Introduction to API

- Set Angle ()



- Applicable to six-axis robots (myCobot 280 series, mechArm series and myCobot 320 series)



- Applicable to four-axis robots (myPalletizer series)



- Parameters:

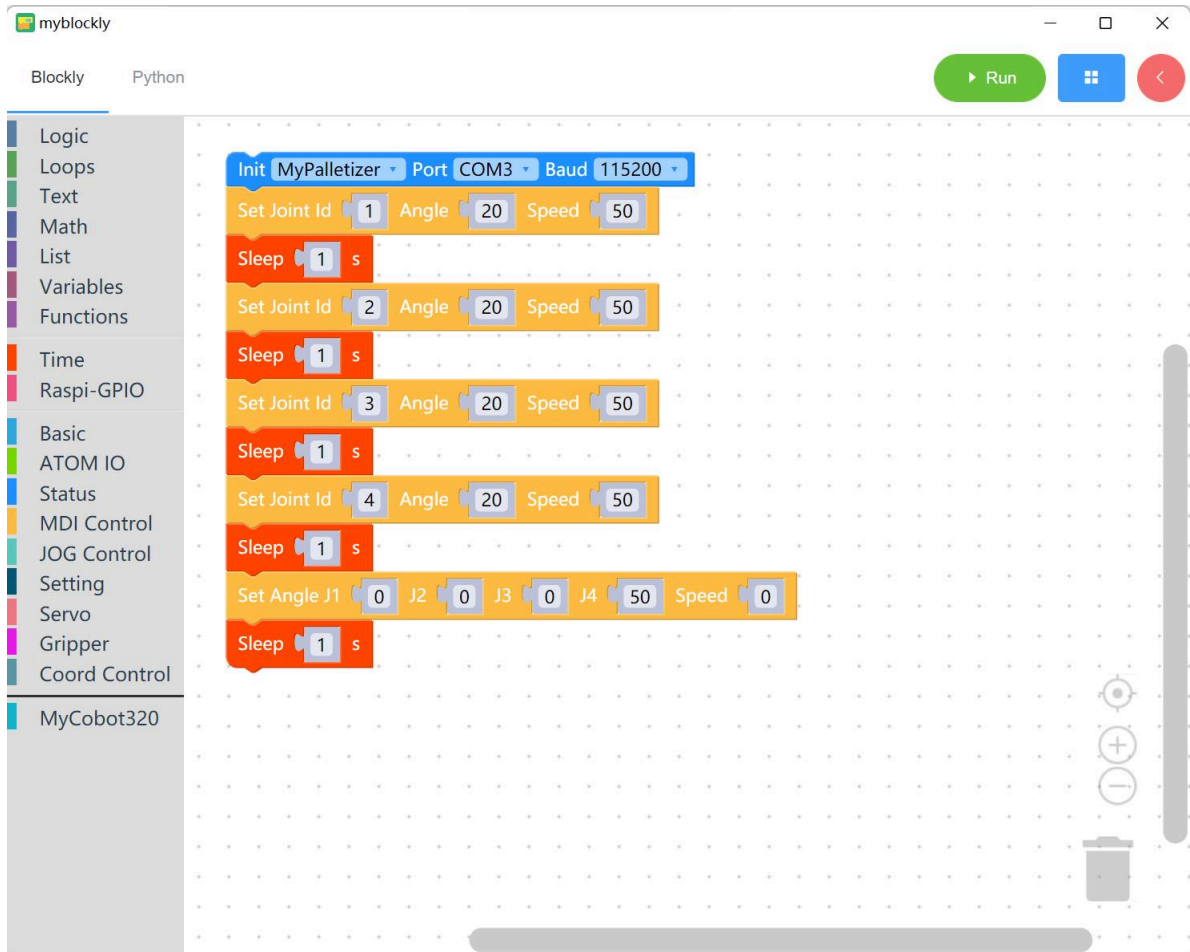
Joint angle: set angles within its due range according to your needs (refer to [2 Products Profile](#))

Speed: set speed within its due range according to your needs (refer to [2 Products Profile](#))

- Function: multiple joints move to corresponding degree at the preset speed.

Simple Demo

- Program for display:



- Motion:

Arms move to the starting point,

after 2 seconds, Joint 1, Joint 2, Joint 3 and Joint 4 move to 30 degree, 30 degree, -30 degree and 50 degree respectively at the speed of 50,

after 2 seconds, arms move to the starting point at the speed of 50,

after 2 seconds, Joint 1, Joint 2, Joint 3 and Joint 4 move to -30 degree, 0 degree, 30 degree and -50 degree respectively at the speed of 50,

program is over in 2 seconds.

Swinging Arms Left and Right

Preparation

M5Stack series: Make sure robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for swinging arms left and right.

Introduction to API

- `power_on()`



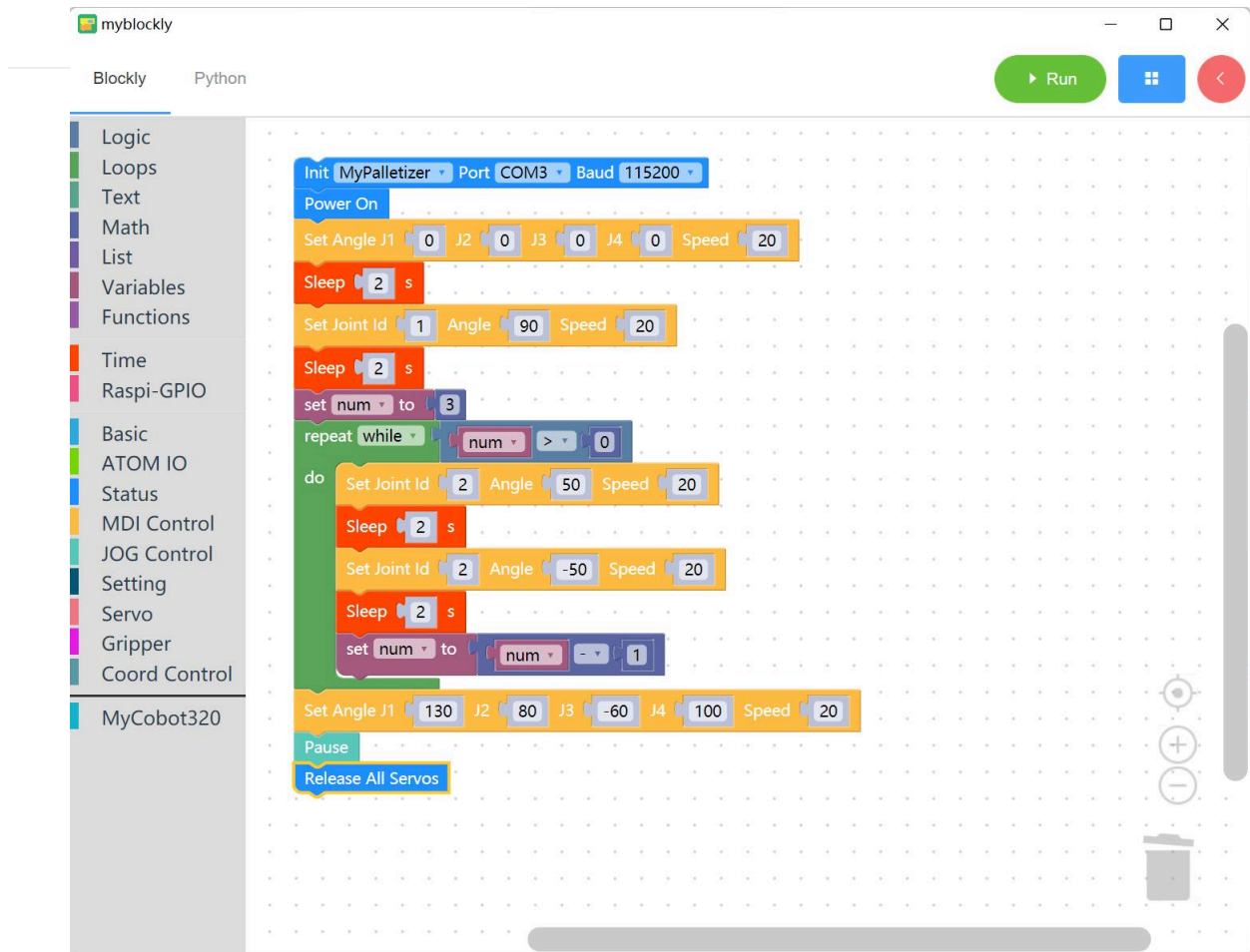
- Applicable to myCobot 280 series, mechArm series, myCobot 320 series and myPalletizer series
- Function: initiate system
- `release_all_servos()`



- Applicable to myCobot 280 series, mechArm series, myCobot 320 series and myPalletizer series
- Function: release all servo motors

Simple Demo

- Program for display:



- Motion:

Initiating system, arms move to starting point, after 2 seconds,

Joint 1 move to 50 degree at the speed of 20, after 2 seconds,

Joint 2 move to -50 degree at the speed of 20, after 2 seconds,

the above process loops again,

Joint 1, Joint 2, Joint 3 and Joint 4 move to 130 degree, 80 degree, -60 degree and 100 degree respectively.

Motion stops, and then all servos release.

Let Robot Dance

Preparation

M5Stack series: Make sure robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

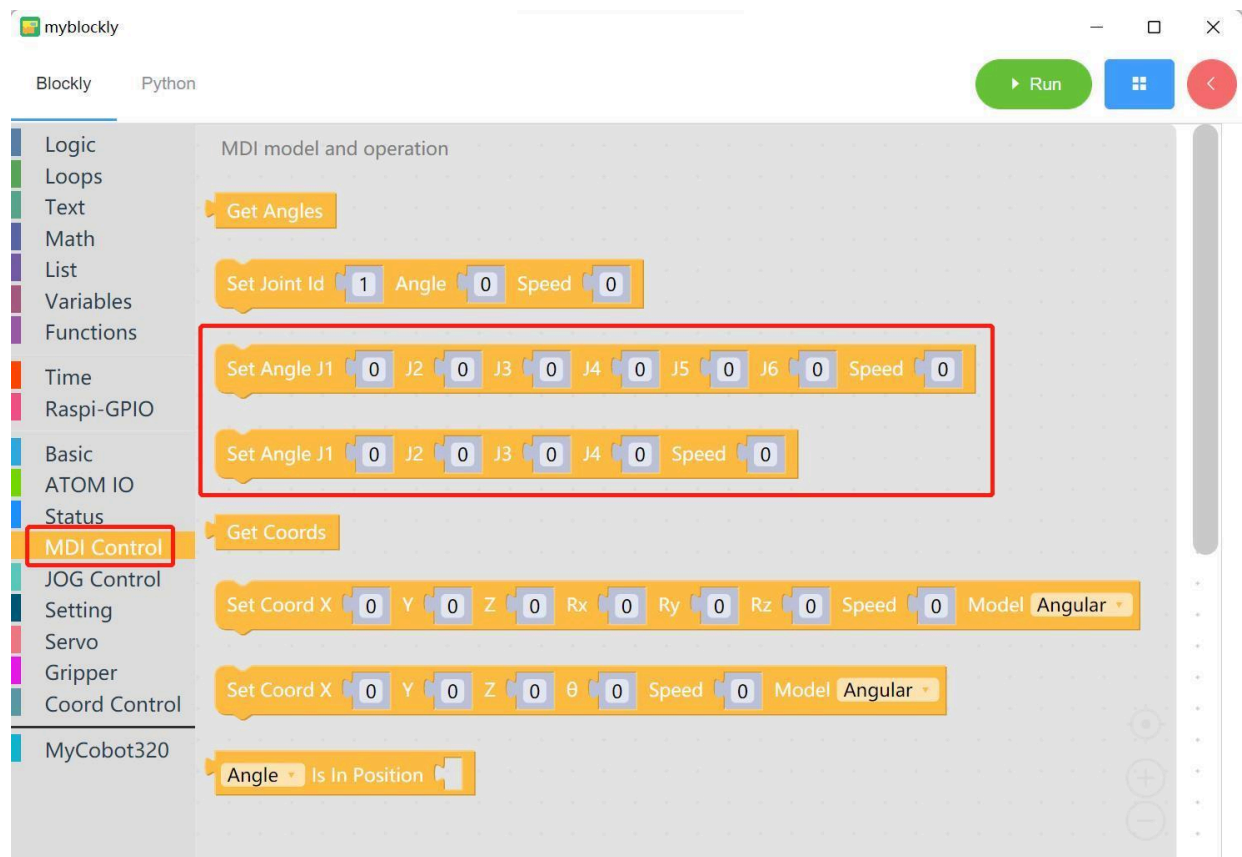
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for swinging arms left and right.

Introduction to API

- Set Angle



- Applicable to six-axis robots (myCobot 280 series, mechArm series and myCobot 320 series)



- Applicable to four-axis robots (myPalletizer series)



- Parameters:

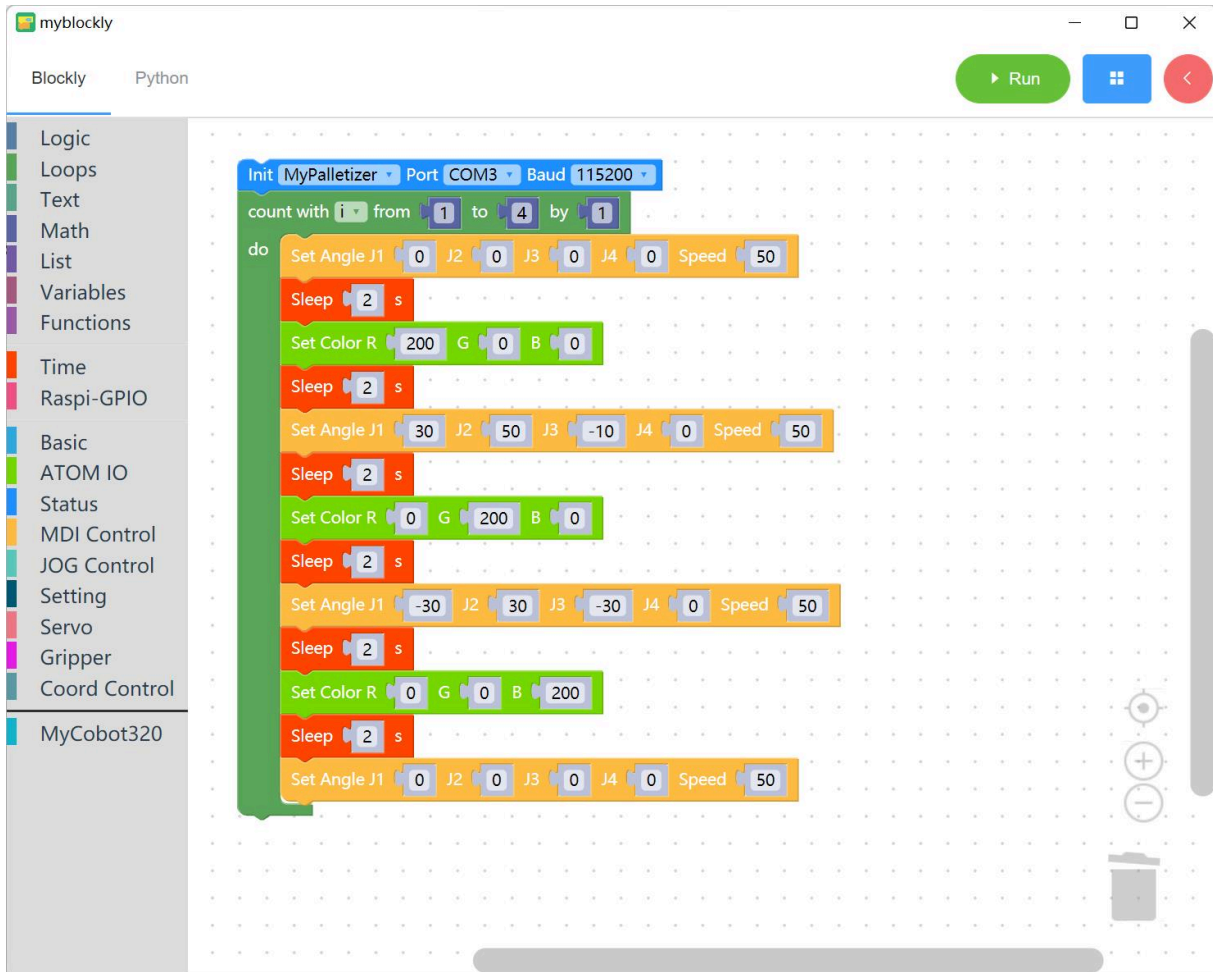
Joint angle: set angles within its due range according to your needs (refer to [2 Products Profile](#))

Speed: set speed within its due range according to your needs (refer to [2 Products Profile](#))

- Function: multiple joints move to corresponding degree at the preset speed.

Simple Demo

- Program for display:



- Motion:

All arms move to starting point, RGB light changes to red,

after 2 seconds, Joint 1, Joint 2 and Joint 3 move to 30 degree, 50 degree and -10 degree at the speed of 50 respectively,

after 2 seconds, RGB light changes to green,

after 2 seconds, Joint 1, Joint 2 and Joint 3 move to -30 degree, 30 degree and -30 degree at the speed of 50 respectively,

after 2 seconds, RGB light changes to blue,

after 2 seconds, all arms move to starting point.

The Use of Gripper

Preparation

M5Stack series: Make sure the robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

Other series: Make sure the robot is in normal status.

This section takes myPalletizer 260 M5Stack as an example to explain the use of gripper. Go to [2.8 Accessories](#) for more information about gripper.

Purpose for this section

This section introduces instructions for using gripper.

Introduction to API

- `set_gripper_ini()`



- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Function: Set the current position to zero

- `set_gripper_state(flag, speed)`



- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
 - flag (int): 0 means open, and 1 means close
 - speed (int): range from 0 to 100
- Function: set gripper switch state

- `set_gripper_value(value, speed)`



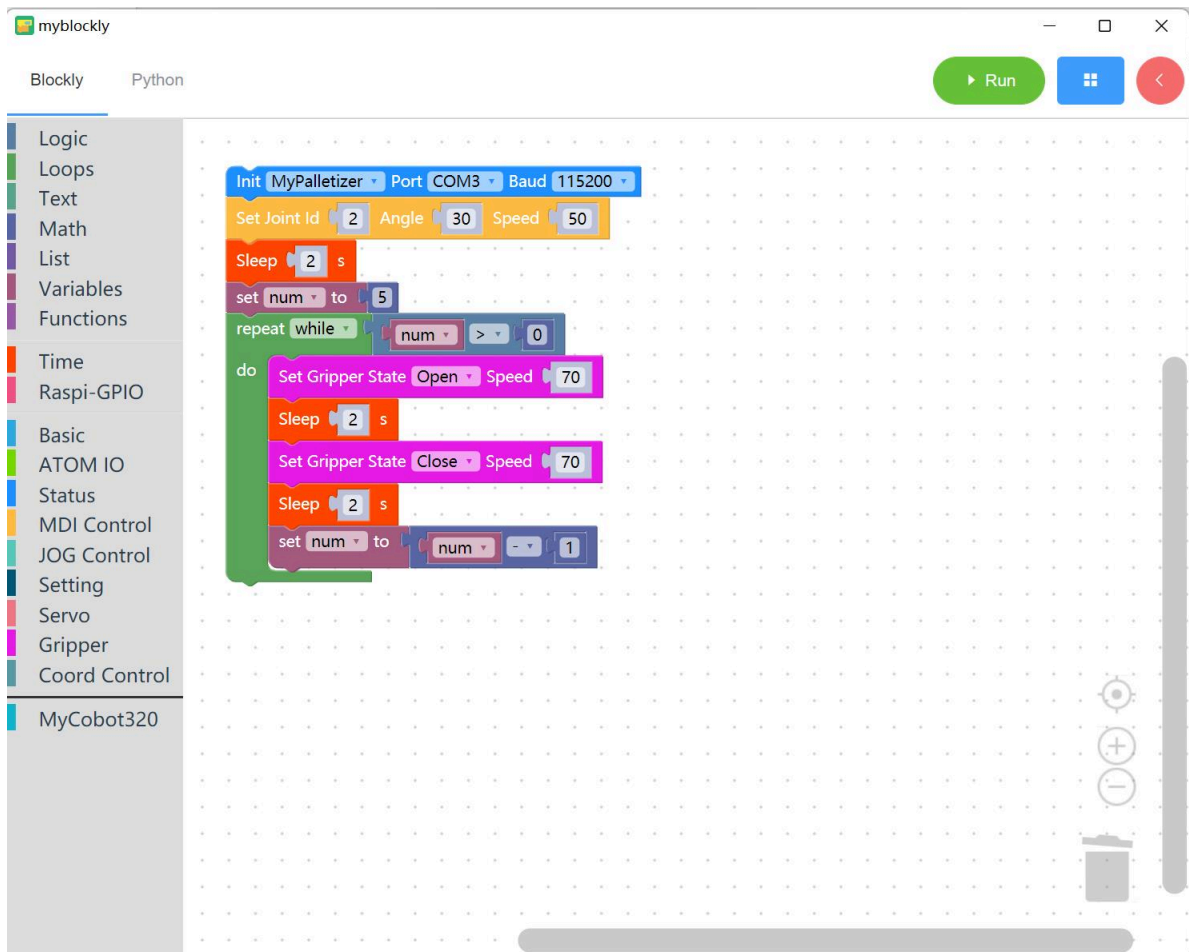
- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
 - Parameter:
 - `value()` : range from 0 to 100
 - `speed()` : range from 0 to 100
 - Function: set gripper value
- `is_gripper_moving()`

Is Gripper Moving

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Function: Judge whether the gripper is moving or not

Simple Demo

- Program for display



- Motion:
 - Joint 2 move to 30 degree at the speed of 50,
 - after 2 seconds, gripper opens at the speed of 70,
 - after 2 seconds, gripper closes at the speed of 70,
 - the process loops 5 times.

The Use of Sucking Pump

Preparation

M5Stack series: Make sure robot is connected with PC (Go to [5.1 myBlockly](#) for more information).

Other series: Make sure the robot is in normal status.

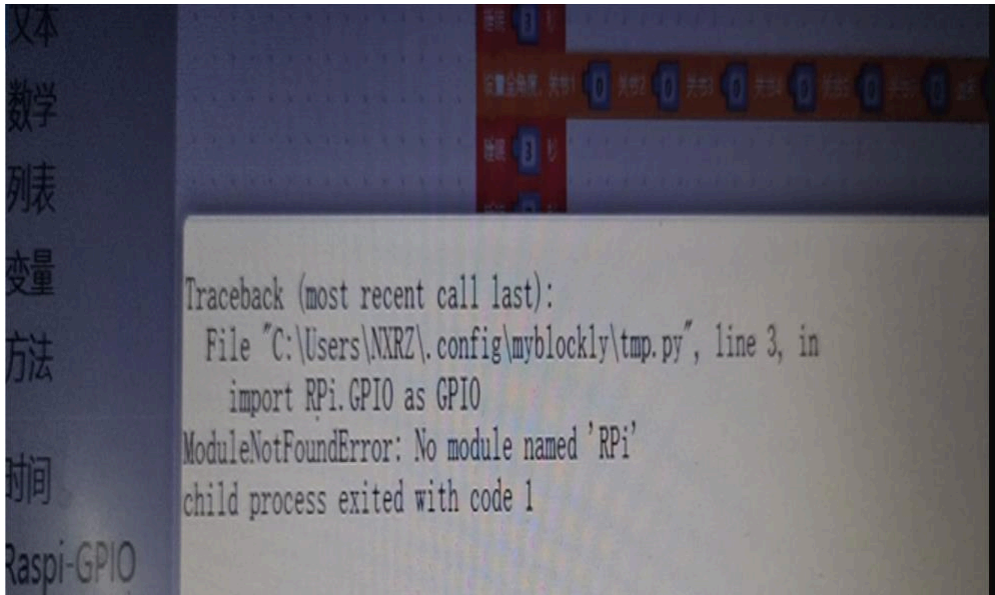
This section takes myPalletizer 260 M5Stack as an example to explain the use of suction pump. Go to [2.8 Accessories](#) for more information about sucking pump.

Purpose for this section

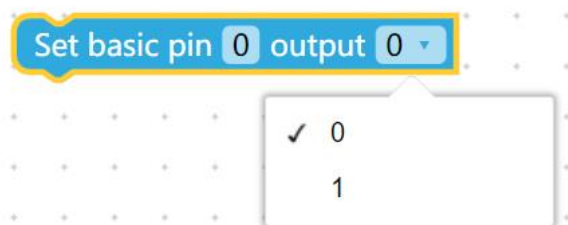
This section introduces instructions for using sucking pump.

Introduction to API

Notice: M5Stack version is unable to use the blocks belonging to Raspberry Pi version. Otherwise, the system may report an error.



- Set Basic pin () output () (for M5Stack version)



- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
 - pin () : the numerical part of the numbers marked at the bottom of the equipment
 - output () : 0 means setting to the running state, and 1 means setting to the stop state

- Function: set the working state of the preset bottom pin

- `get_basic_input(pin_no)` (for M5Stack version)

Get basic pin 0 input

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
 - `pin()` : the numerical part of the numbers marked at the bottom of the equipment
- Function: get the working state of the bottom pin number
- `set mode ()` (for Raspberry Pi version)

Set mode BCM

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
 - `mode()` : "BCM" or "BOARD"
- Function: set Raspberry Pi GPIO Pin Mode
- `set pin() mode()` (for Raspberry Pi version)

Set pin 0 mode OUT

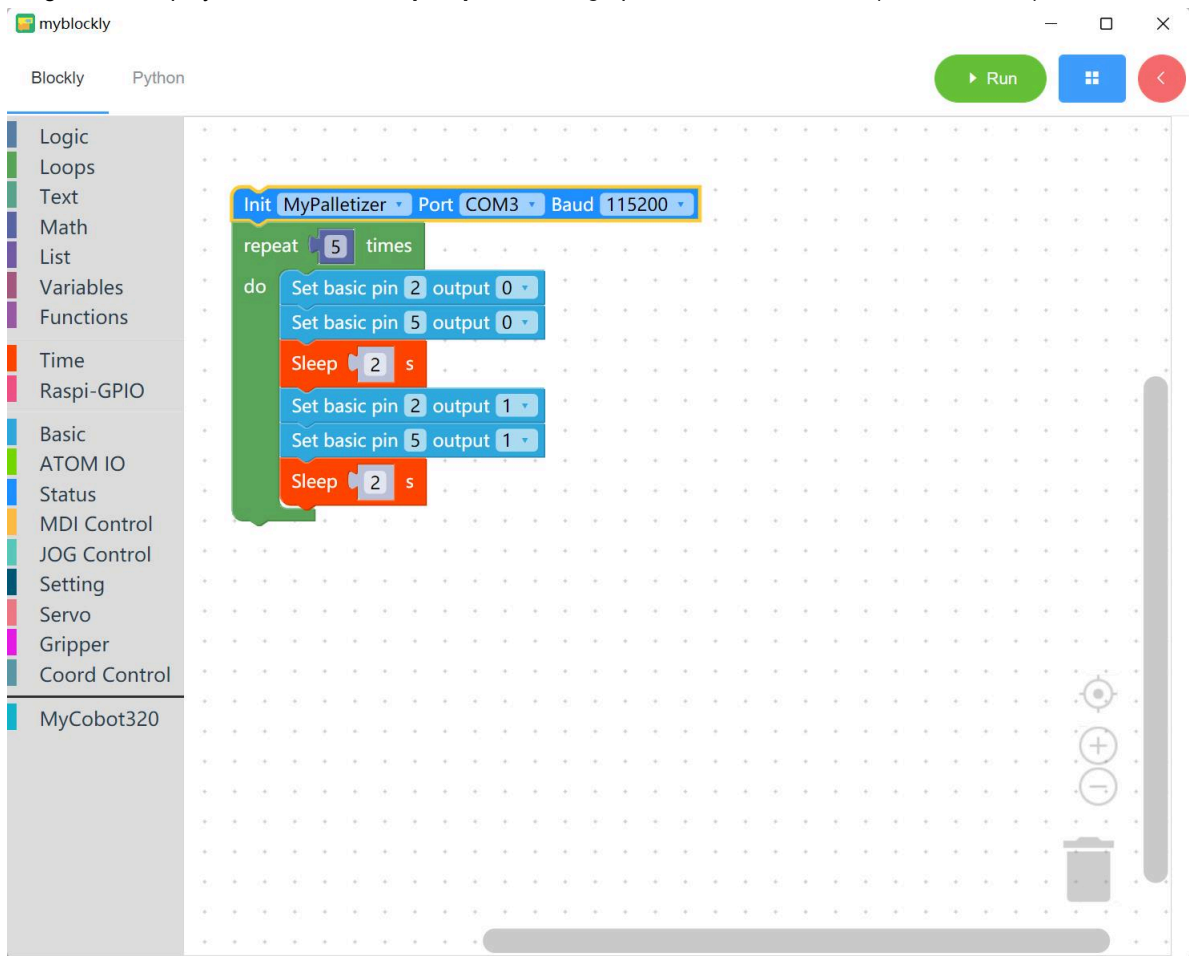
- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
 - `pin ()` : the numerical part of the numbers marked at the bottom of the equipment
 - `mode ()` : `in` means signal import, and `OUT` means signal output
- Function: set signal of the preset bottom pin
- `set pin() output()` (for Raspberry Pi version)

Set pin 0 output HIGH

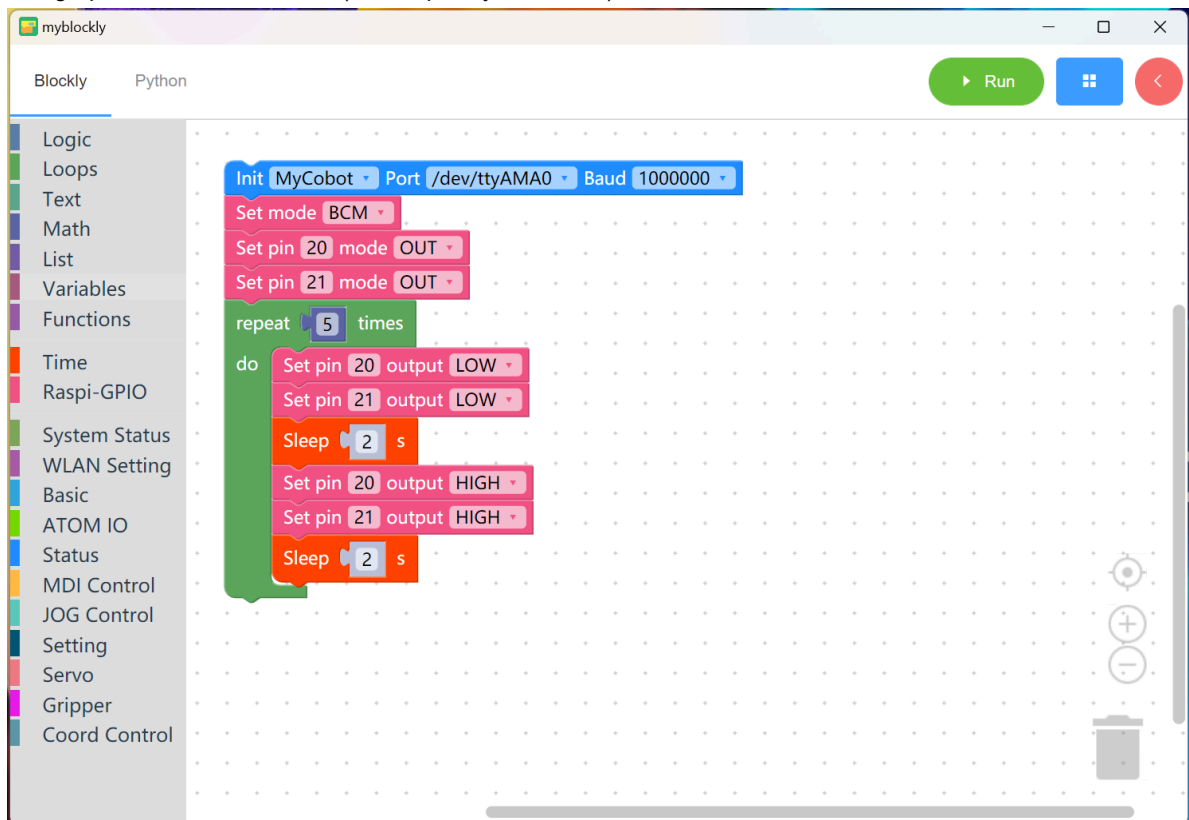
- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
 - `pin()` : the numerical part of the numbers marked at the bottom of the equipment
 - `output()` : `HIGH` means high level of sucking pump working state, and `LOW` means low level of sucking pump working state
- Function: set the working state of bottom pin to high level or low level

Simple Demo

- Program for display: **Vertical suction pump V1.0** The graphic code is as follows: (for M5 version)



The graphic code is as follows: (for Raspberry Pi version)



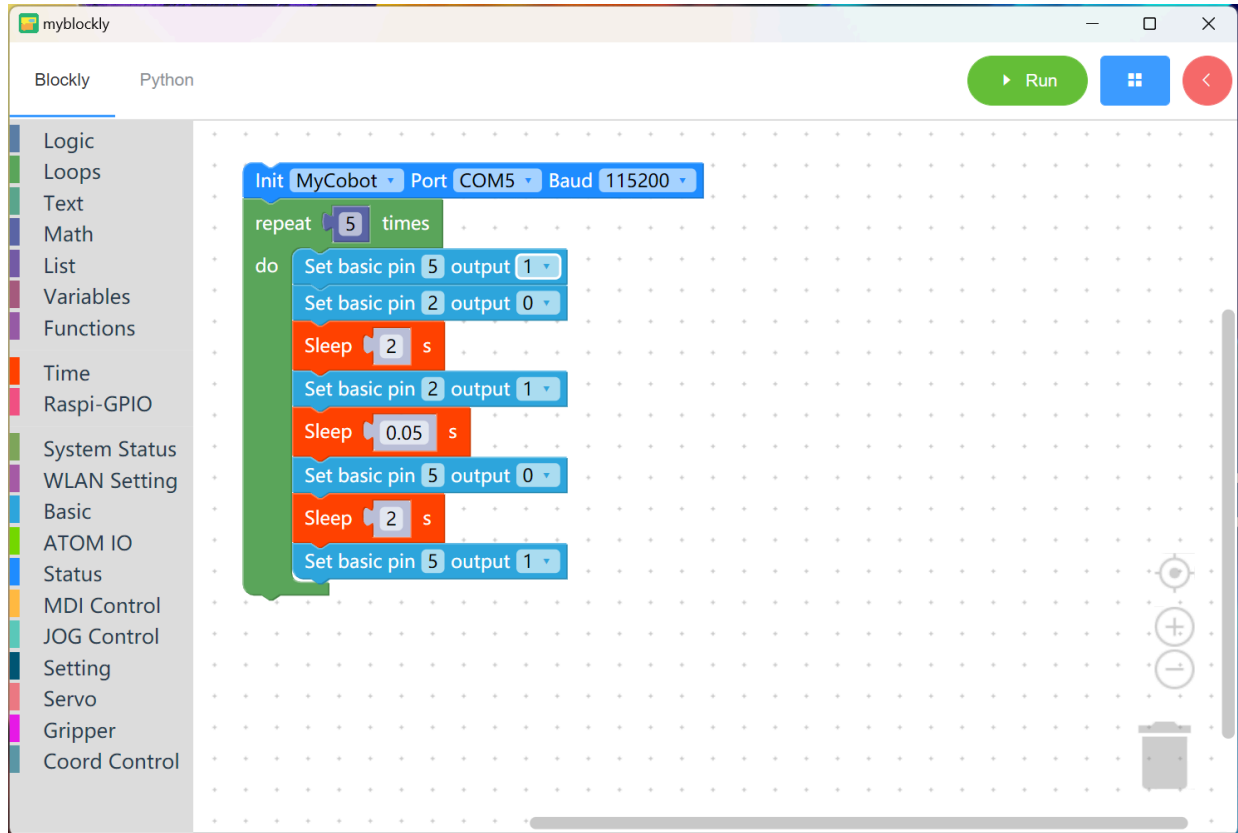
- Motion:

Sucking pump vibrates and work to suck objects,

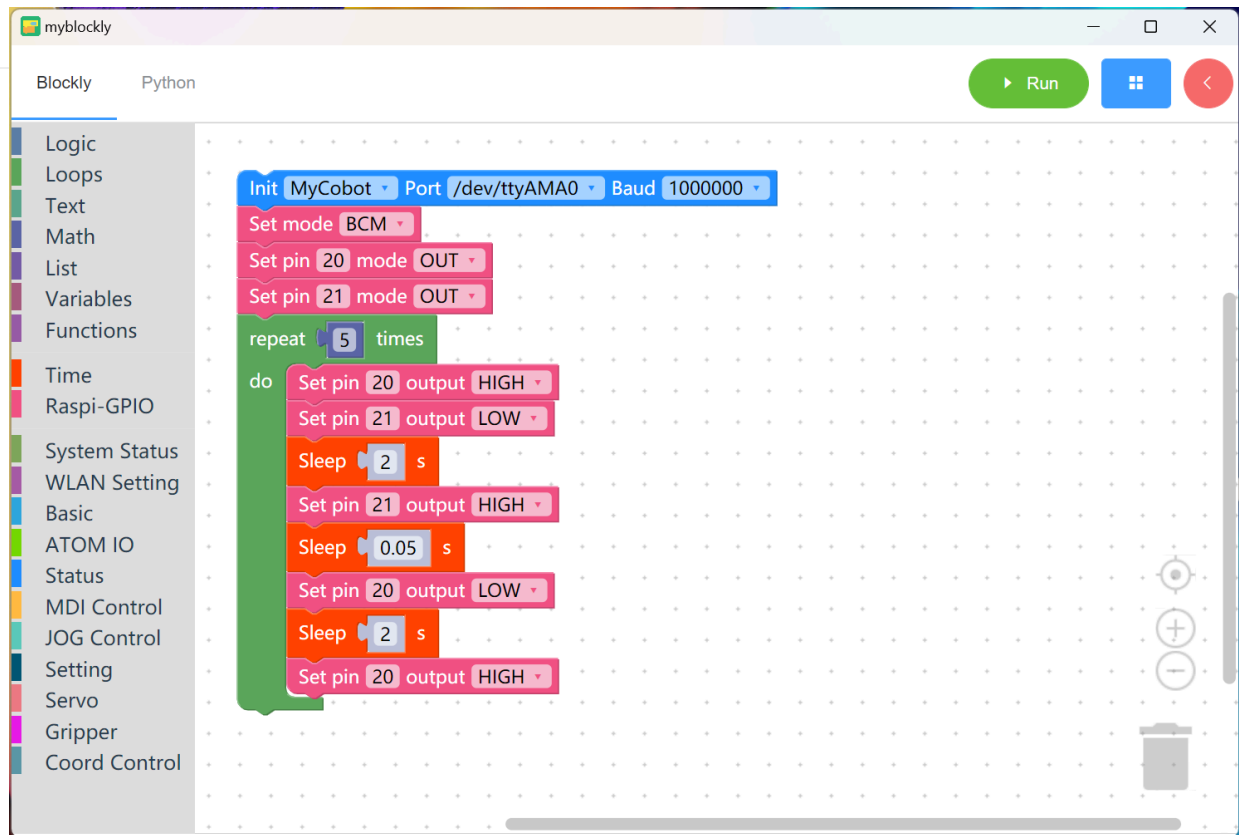
after 2 seconds, sucking pump put objects down,

after 2 seconds, it vibrates again. The whole process loops 5 times.

Vertical suction pump V2.0 *The graphic code is as follows: (for M5 version)*



The graphic code is as follows: (for Raspberry Pi version)



- Motion: First, close the solenoid valve and open the air release valve to prepare for starting the suction pump; After two seconds, close the vent valve. After 0.05 seconds, open the solenoid valve and the suction pump will vibrate, starting to work. The suction pump will pick up an object and after two seconds, put it down. Close the air release valve and repeat the previous action until the program is completed.

Gripper Test

Before you start

M5Stack series: Make sure the robot is connected to the computer (for details, please refer to [5.1myBlockly](#))

Others: Make sure the machine is normal

To use the gripper test function, myblockly needs to be updated to v1.5.5

Learning content in this chapter

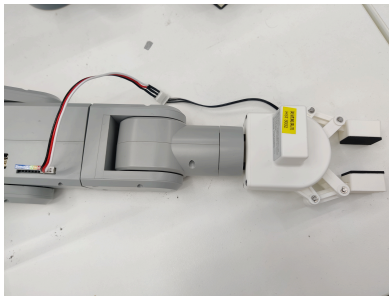
How to use myBlockly to quickly test whether the gripper is normal.

Currently, the gripper test function is only compatible with the following models: myCobot 280 series, mecharm 270 series, myPalletizer 260 series, myArm 300 Pi.

The grippers include adaptive grippers, electric grippers, and pneumatic grippers. Different grippers are compatible with different robot arm models. For details, please refer to [2.7 Accessories](#).

Here, mecharm 270 M5 and adaptive grippers are taken as examples.

First install the gripper on the machine



myblockly connects to the machine serial port

myBlockly

Blockly Python

Logic
Loops
Text
Math
List
Variables
Functions
Time
Raspi-GPIO
System Status
WLAN Setting
Basic
ATOM IO
Status
MDI Control
JOG Control
Setting
Servo
Gripper
Coord Control

Init MechArm Port COM4 Baud

COM4
115200

Close ☒ Open ☐
Auto Complete ☒

Quick Move

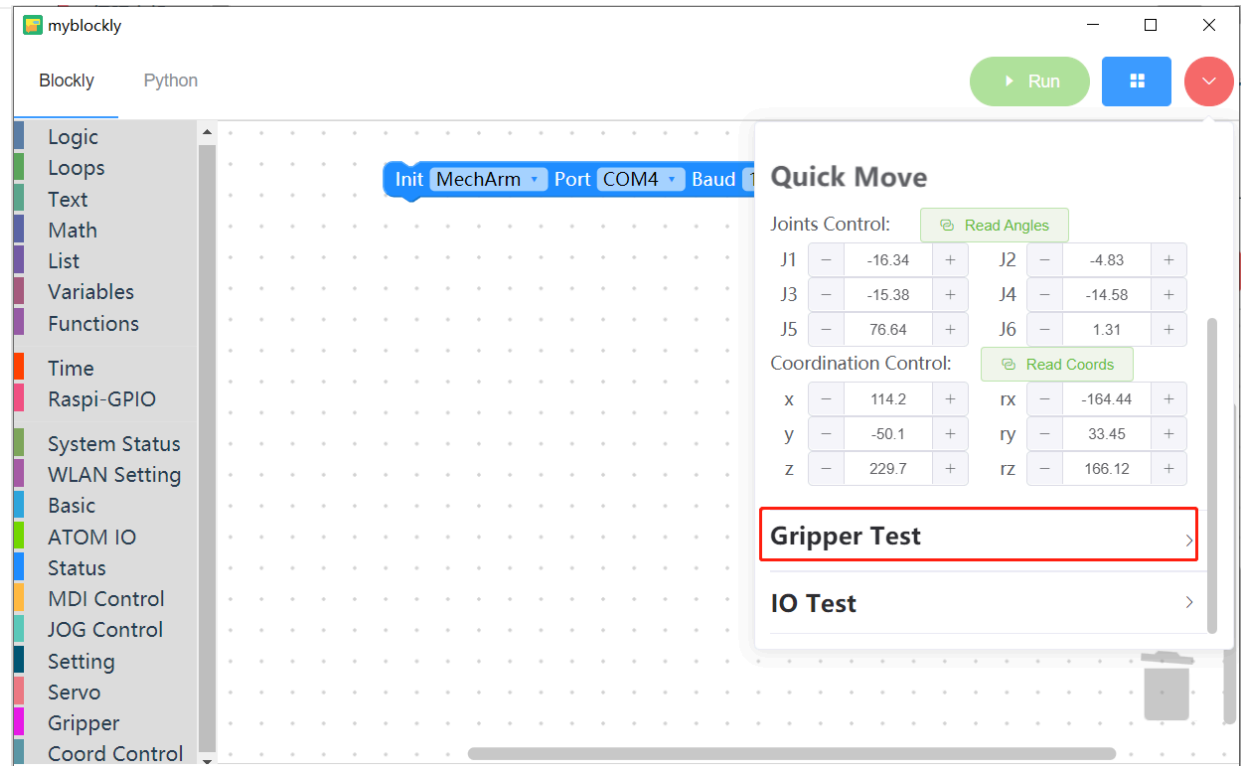
Joints Control: ☒ Read Angles

J1	-	-16.34	+	J2	-	-4.83	+
J3	-	-15.38	+	J4	-	-14.58	+
J5	-	76.64	+	J6	-	1.31	+

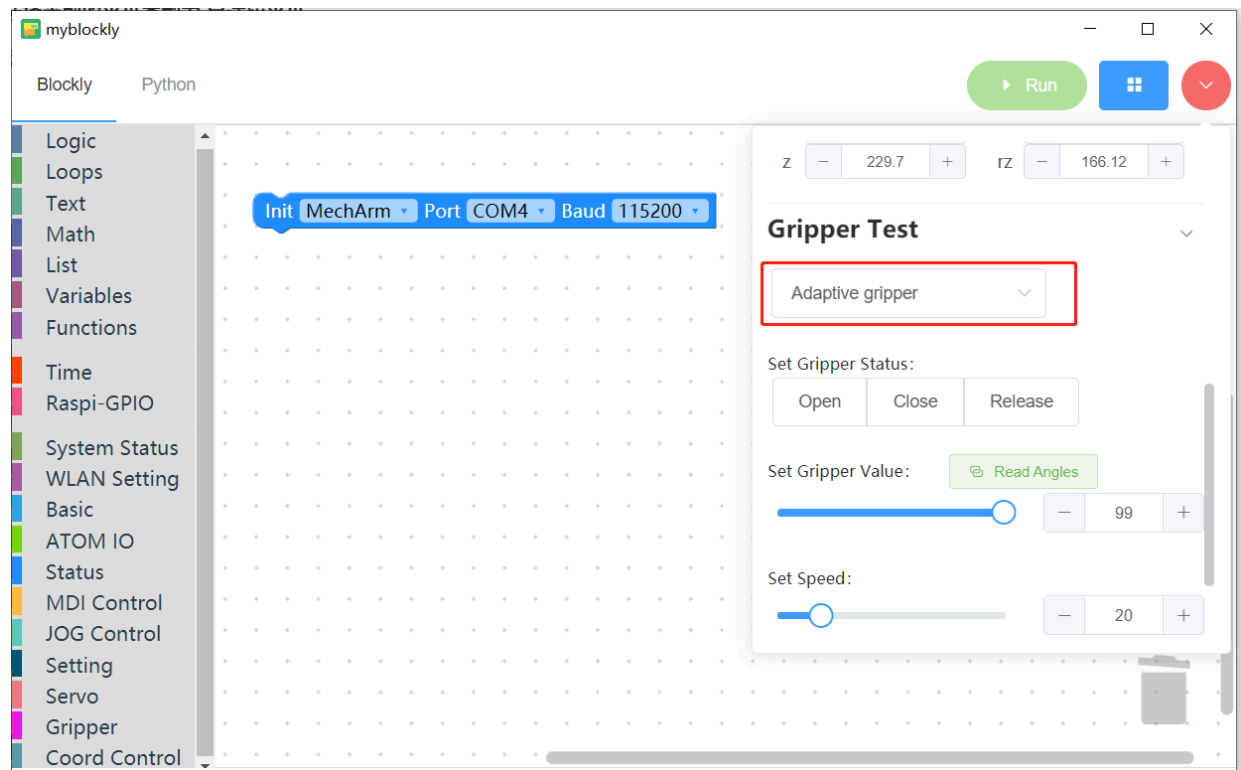
Coordination Control: ☒ Read Coords

x	-	114.2	+	rx	-	-164.44	+
y	-	-50.1	+	ry	-	33.45	+
z	-	229.7	+	rz	-	166.12	+

Open Gripper Test



Select the current gripper type as Adaptive gripper



Then you can try the following:

- Set the gripper state: Open, Closed, Released. Pay attention to the gripper movement.
- Click the Read Angles button to get the current gripper value.
- You can fine-tune the gripper value.
- You can set the gripper movement speed

■ Tips: When the gripper type is Dexterous Hand, the gripper value cannot be set, only its status can be set.

IO test

Preparation before starting

M5Stack series: Make sure that the robotic arm is connected to the computer (please refer to [\[5.1Myblockly\]](https://docs.elephantrobotics.com/docs/gitbook/5-proGamingApplication-Myblockly-UIFL-UIFL OW-MIND/5.1-Myblockly/) (<https://docs.elephantrobotics.com/docs/gitbook/5-proGamingApplication-Myblockly-UIFL-UIFL OW-MIND/5.1-Myblockly/>))

Others: Make sure the machine is normal

To use the IO test function, myblockly needs to be updated to V1.5.5

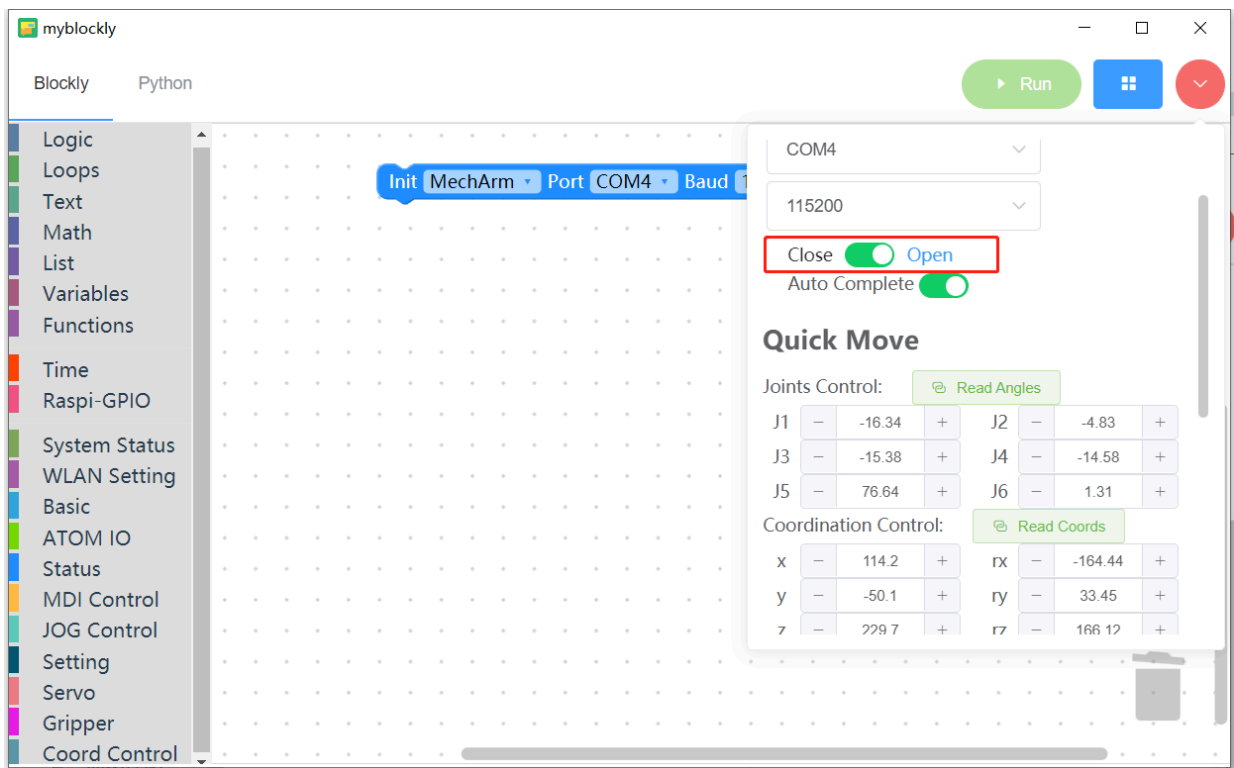
Learning content in this chapter

How to quickly test whether IO is normal to use myblockly.

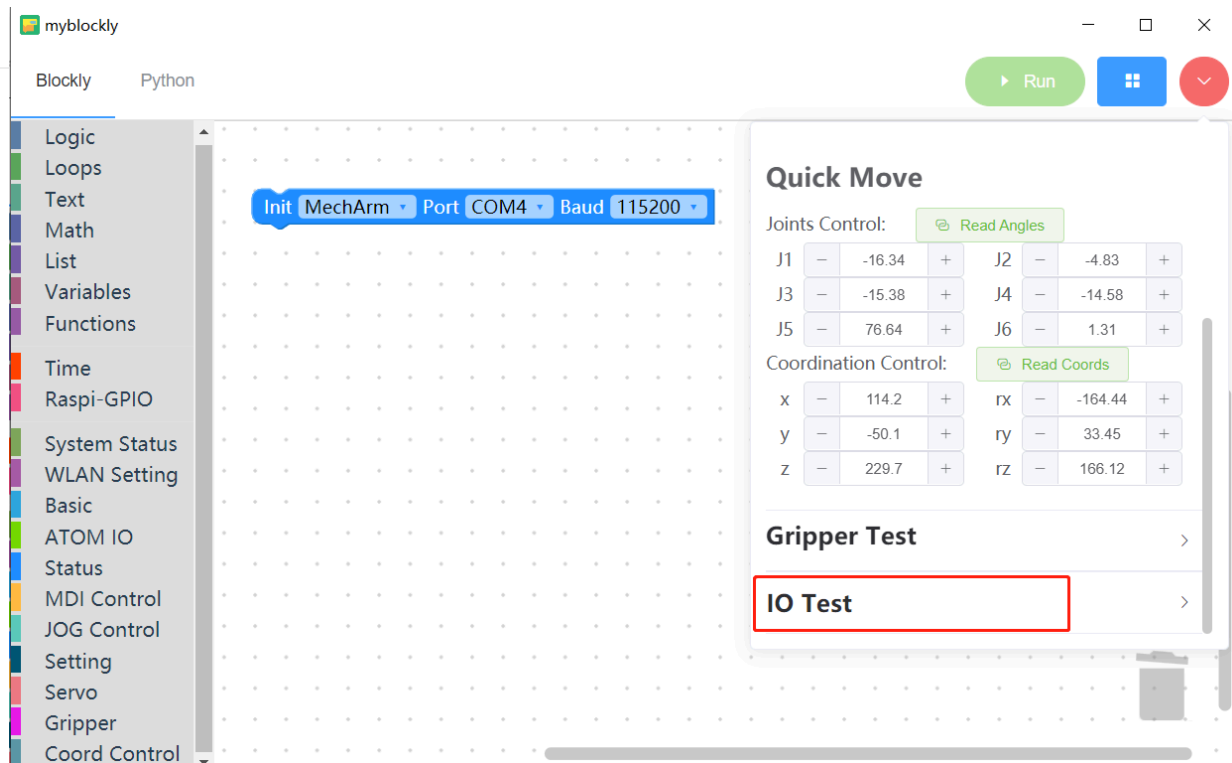
At present, the io test function only adapts to the following models: MyCobot 280 series, MeCharm 270 series, MyPalletizer 260 series, MyARM 300 PI.

Take Mecharm 270 M5 as an example.

myblockly connect the machine serial port

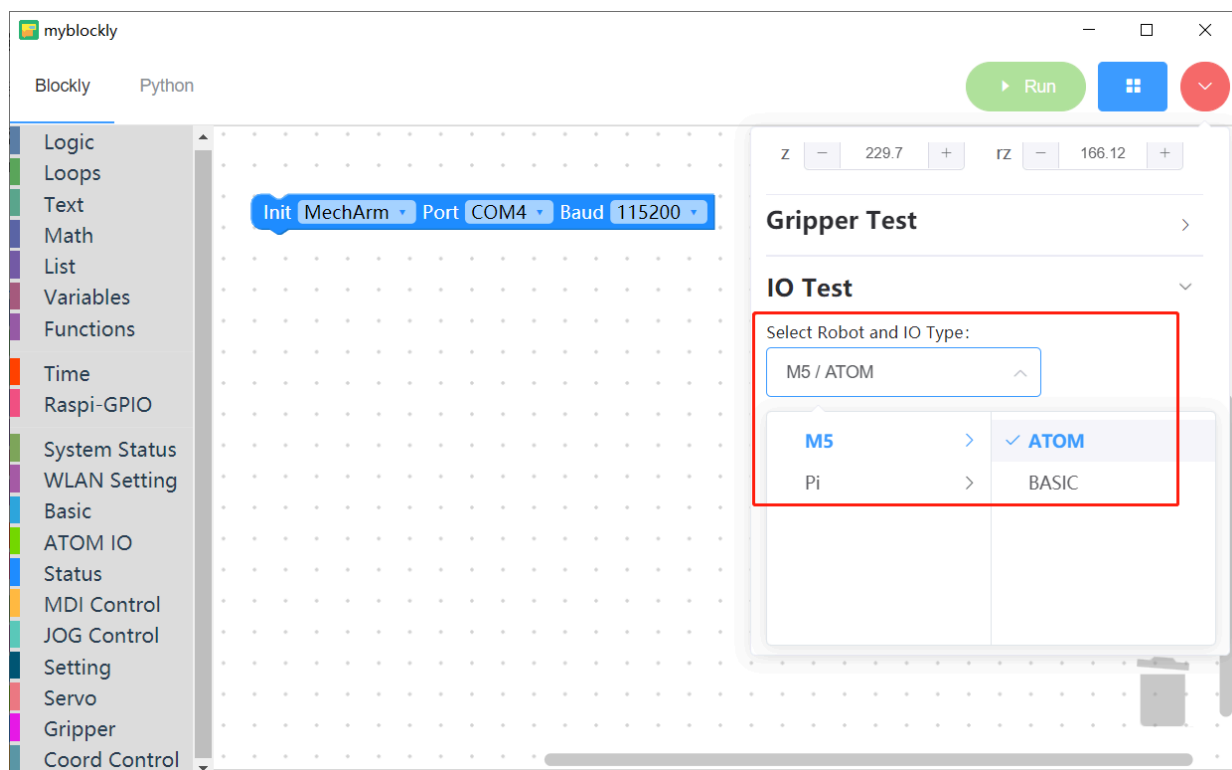


Open IO Test



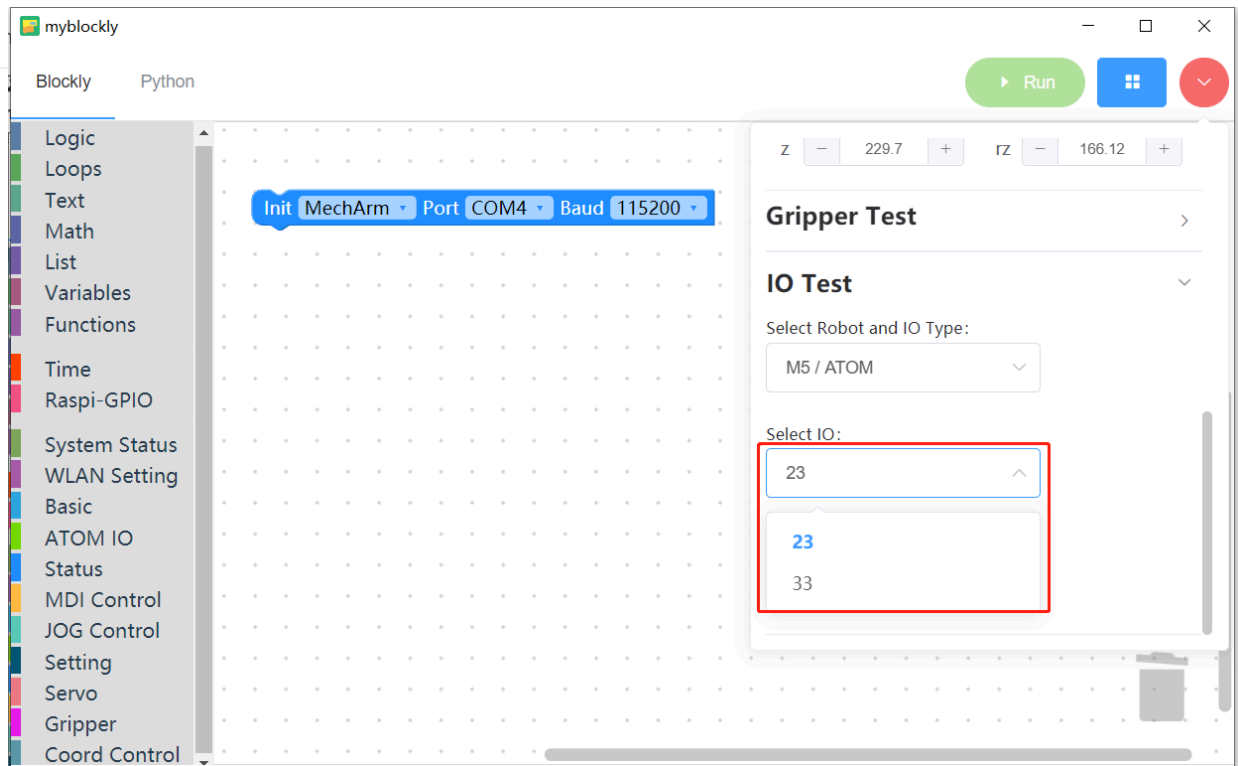
Select IO type

Our current models are M5 models, so choose the M5 type. ATOM and BASIC in the M5 classification can be selected. Here we choose atom



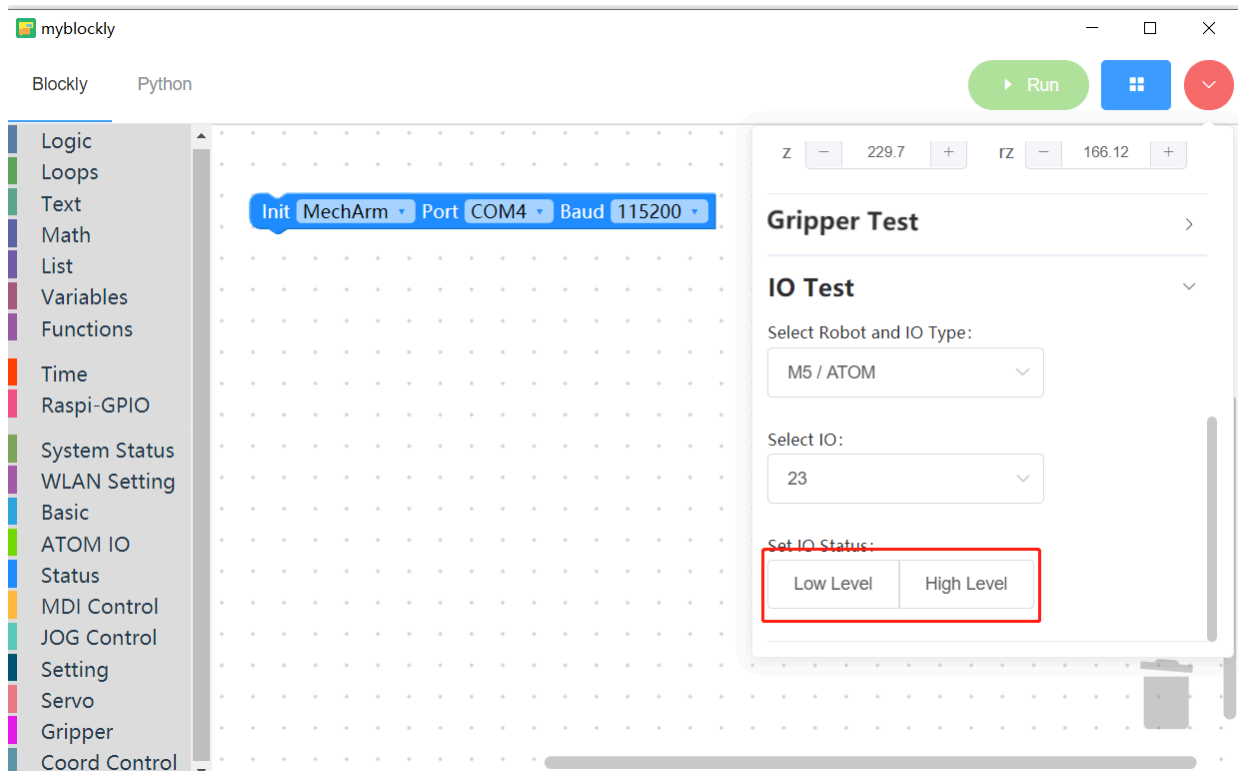
Select ATOM IO

meacharm 270 m5 Atom IO port is currently only 23 and 33 open

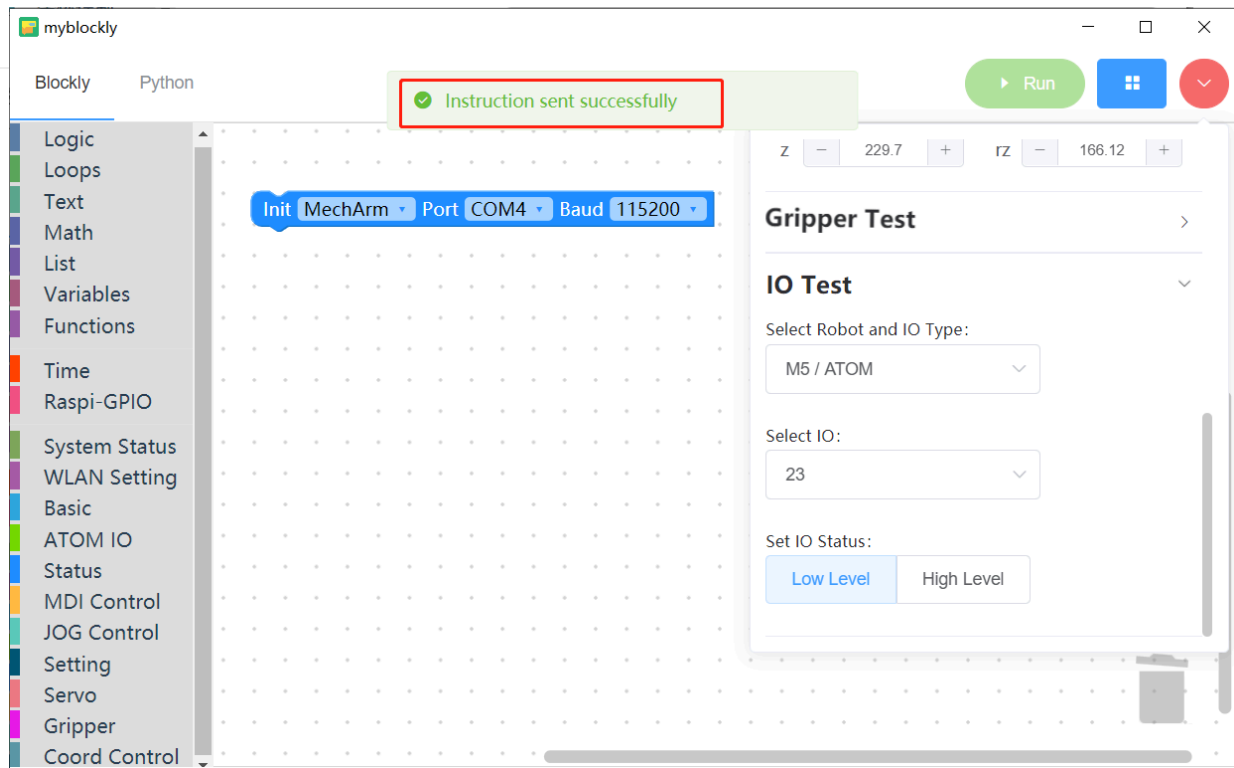


Change IO Status

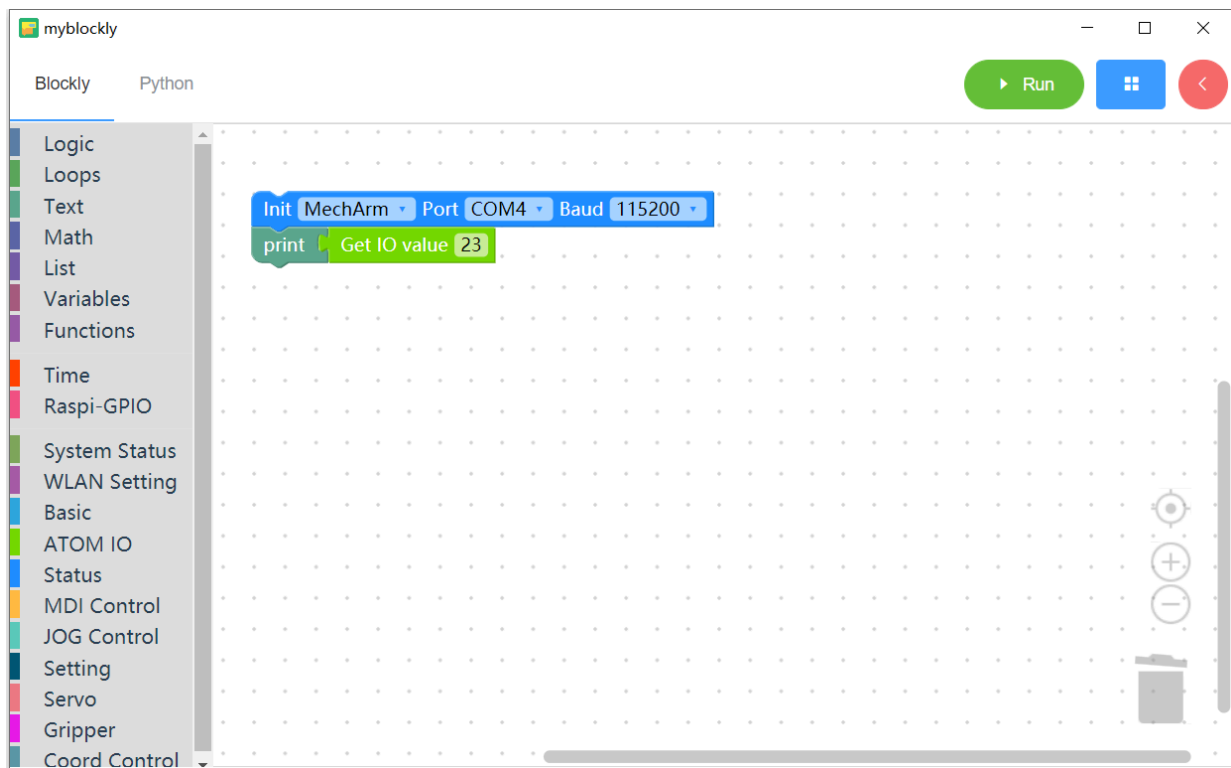
You can click the **Low Level** **High Level** button to change the current selected IO port status



Set the high and low level instructions to send successfully



After changing the IO status, you can read the atom IO level through the building block to see if it is successful



What is Python?

Our products are friendly to python and also becomes increasingly perfect for the development of a python API library. Through python, the joint angle, coordinates, gripper and other aspects of the robot can be controlled, and there are many options available. If you want to control our robot arms via Python programming, you are recommended to learn this chapter.



Python was designed in the early 1990s by Guido van Rossum of the Netherlands Society for Mathematics and Computer Science as an alternative to a language called ABC.

Python not only provides efficient, advanced data structures, but also can be used to do simple and effective object-oriented programming.

The Syntax and dynamic typing of **Python** as well as the nature of interpreted languages, make it become a programming language for scripting and rapid application development on most platforms. With the continuous updating of the version and the addition of new features, it is gradually used to develop independent, large-scale projects.

The interpreter of **Python** is easily extensible, and new functions and data types can be extended using the C or C++ language (or other languages that can be called through C language).

Python can also be used for extending program languages in customizable software. **Python** has rich standard libraries and provides source or machine codes suitable for each major system platform.

Installing Python



- **Python** Python's official downloading address: <https://www.python.org/downloads/>
- **Python** downloading and Installation Tutorial for reference only

Applicable equipment:

- myCobot 280
 - myCobot 280 M5
 - myCobot 280 PI
 - myCobot 280 Jetson Nano

- myCobot 280 for Arduino
-

- myCobot 320
 - myCobot 320 M5
 - myCobot 320 PI
- myPalletizer 260
 - myPalletizer 260 M5
 - myPalletizer 260 PI
- mechArm-270
 - mechArm-270 M5
 - mechArm-270 PI

Preconditions for use:

- **M5** series version, the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)
- **Pi \ jetsonnano** series, **ATOM** burns the latest version of **atomMain** (factory default already burnt)

Environment Building

pymycobot is a Python package used for serial communication with myCobot. It supports Python2, Python3.5 and later versions.

Before using pymycobot, make sure to build a Python environment. Follow the steps below to install Python.

1 Download and Installation of Python

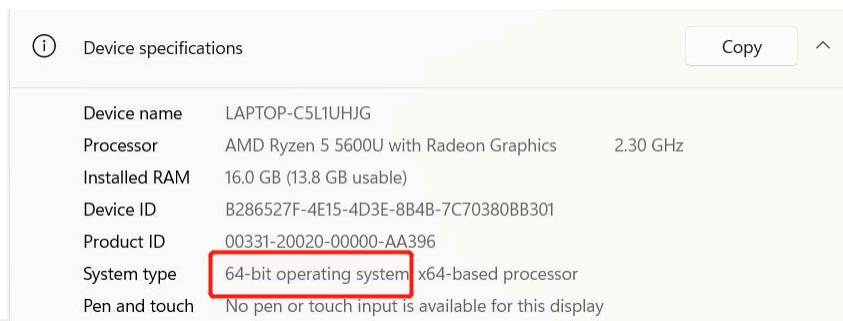
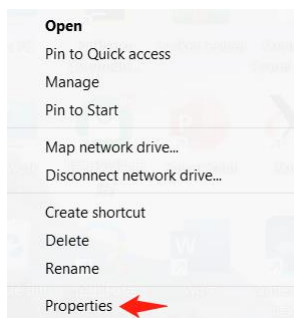
Python is applicable to:

- myCobot 280:
 - myCobot 280 M5
 - myCobot 280 PI
 - myCobot 280 Jetson Nano
 - myCobot 280 for Arduino
- myCobot 320:
 - myCobot 320 M5
 - myCobot 320 PI
- myPalletizer 260:
 - myPalletizer 260 M5
 - myPalletizer 260 PI
- mechArm-270:
 - mechArm-270 M5
 - mechArm-270 PI

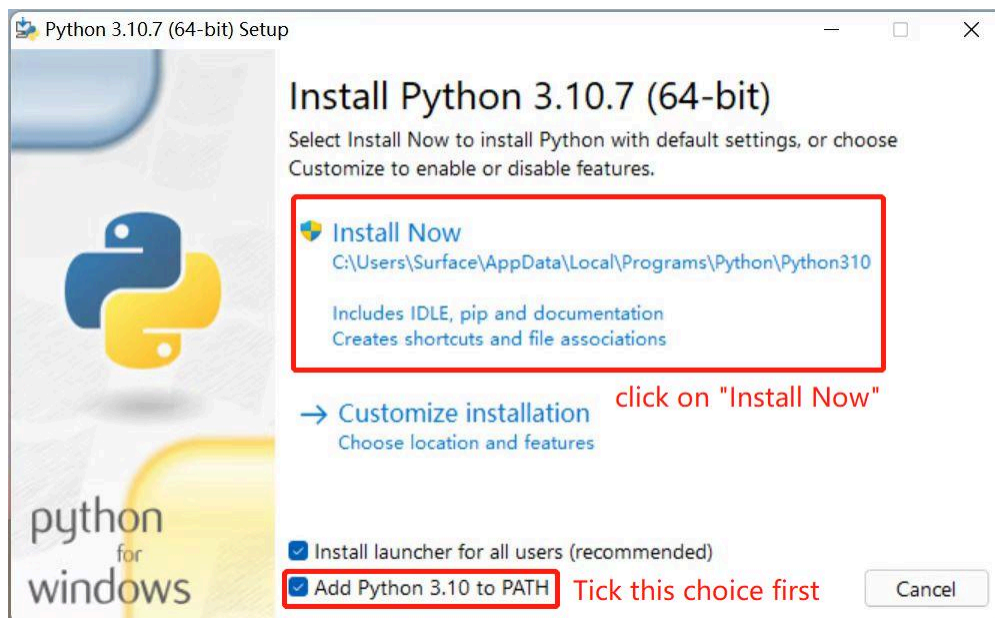
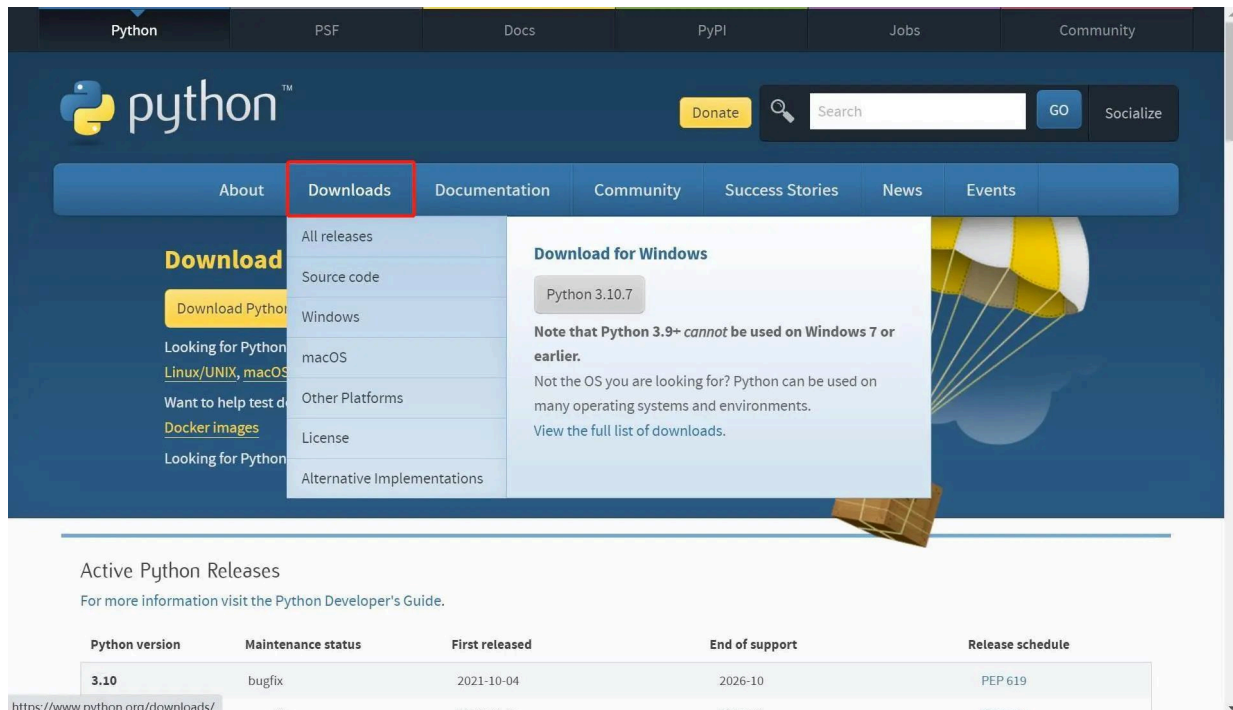
At present, Python has two versions: 2.x and 3.x. These two versions are incompatible with each other. This section takes the version 3.x as an example due to its increasing popularity.

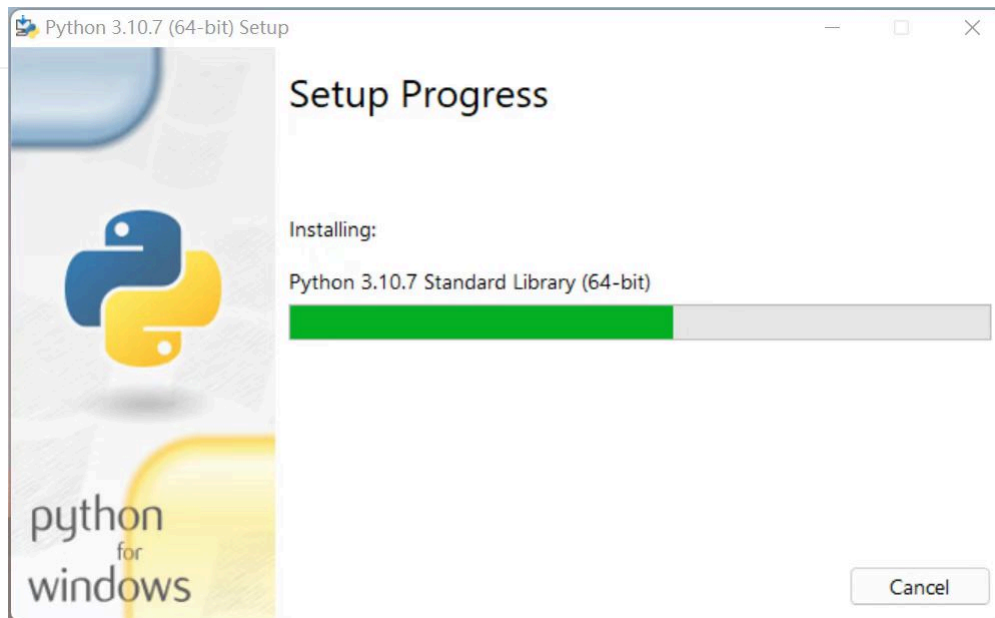
1.1 Installing Python

Notice: Before installation, check the operation system of PC. Press right button on the **My Computer** icon and then select **Properties**. Install the corresponding Python.

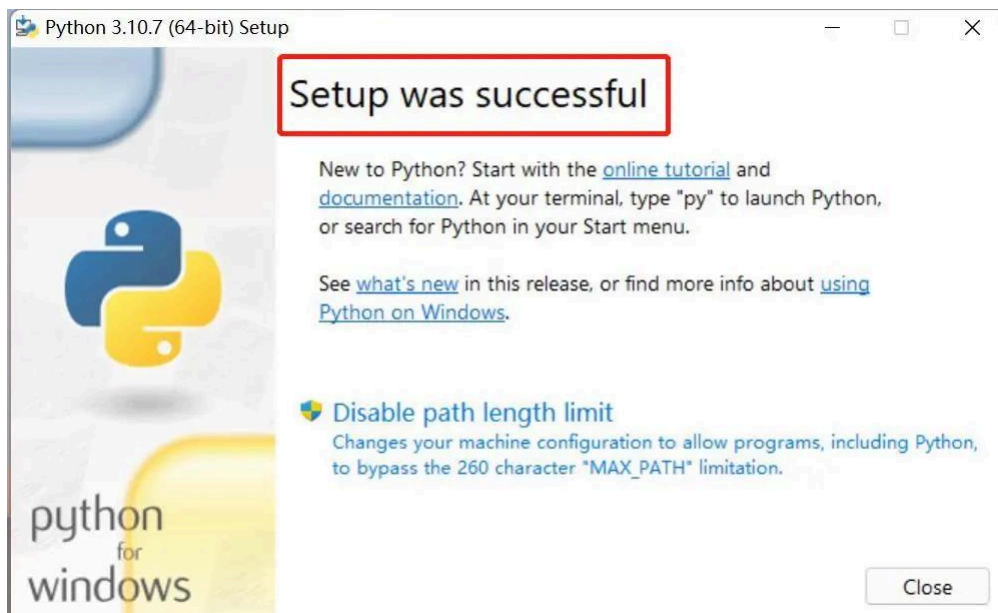


- Go to <http://www.python.org/download/> to download Python.
- Click on **Downloads**, and then download begins. Tick **Add Python 3.10 to PATH**. Click on **Install Now**, and then installation begins.





- **Download and installation complete.**



1.2 Running Python

Open the command prompt window (Win+R, input `cmd` and press `Enter`). Type `Python` .

Successful Installation:

```

C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [版本 10.0.22000.978]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Surface>python
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

```

This on-screen instruction means that Python is successfully installed. The prompt `>>>` means Python interactive environment. If you input a Python code to get the execution result immediately.

Error Report:

If a wrong instruction is typed, for example "pythonn", the system may report an error.

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\Users\12599>pythonn
'pythonn' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\12599>

```

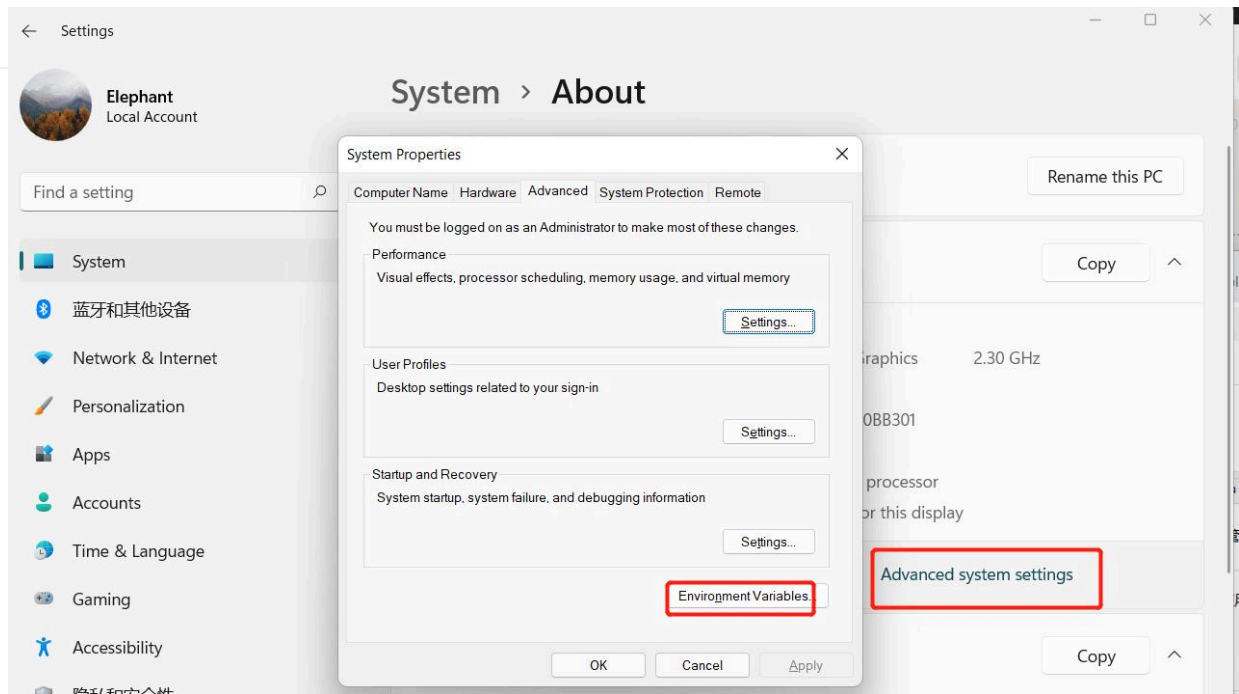
Notice: Generally, the error results from lack of environment configuration. Refer to **1.3 Environment Configuration** to solve problems.

1.3 Environment Variable Configuration

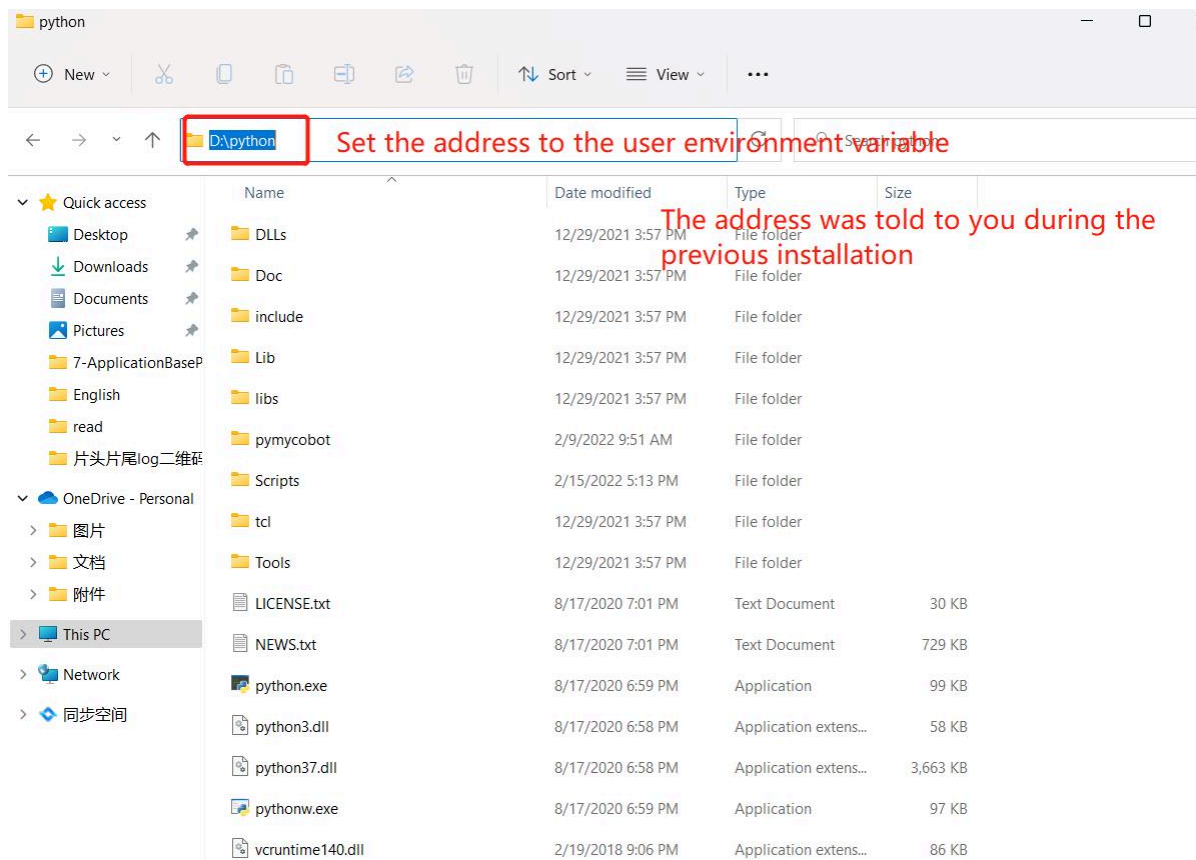
Windows follows the path set by a Path environment variable in search of **python.exe**. Otherwise, an error will be reported. If you fail to tick **Add Python 3.9 to PATH** during installation, you need to manually add the path where python.exe is located into environment variable or download python again. Remember to tick **Add Python 3.9 to PATH**.

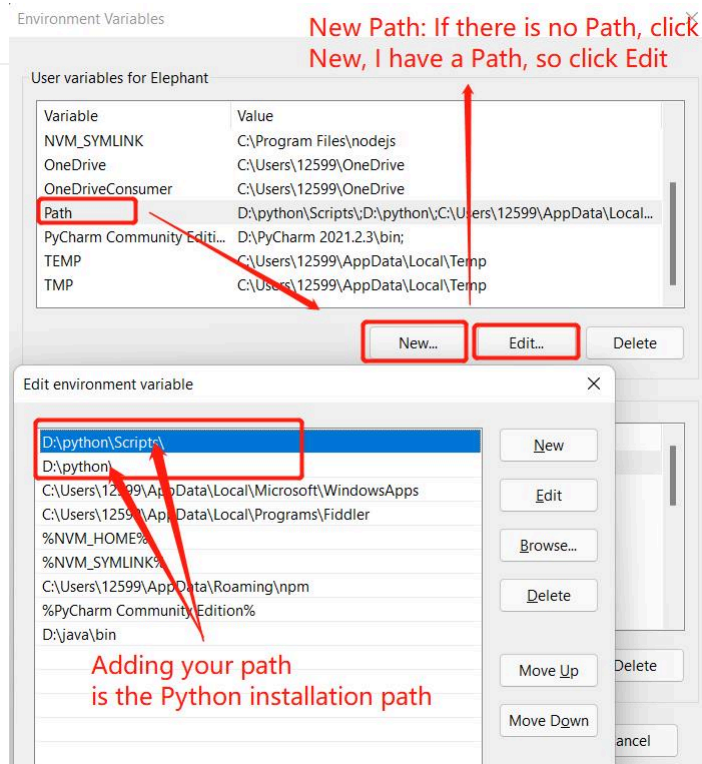
Follow the steps below to add python into environment variable manually.

- Right click on **My Computer** icon --> Properties --> Advanced System Settings --> Environment Variables

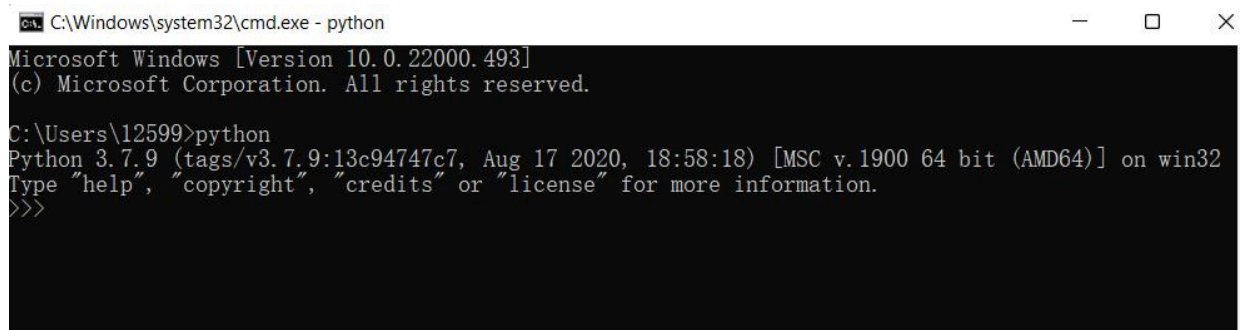


- The environment variables include user variables and system variables. For user variables, users can utilize their own downloaded programs via `cmd` command. Write the absolute path of the target program into the user variables.





- After the configuration, open the command prompt window (Win+R; input `cmd` and press `Enter`), and type `Python`.



2 Installation of PyCharm

PyCharm is a powerful python editor with the nature of cross-platform. Follow the steps below to download and install PyCharm.

Go to [PyCharm](#) to download PyCharm.

2.1 Download and Installation

Official website view:



Version: 2022.2.3
Build: 222.4345.23
11 October 2022

[System requirements](#)

[Installation instructions](#)

[Other versions](#)

[Third-party software](#)

Download PyCharm

[Windows](#)

[macOS](#)

[Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free 30-day trial available

Community

For pure Python development

[Download](#)

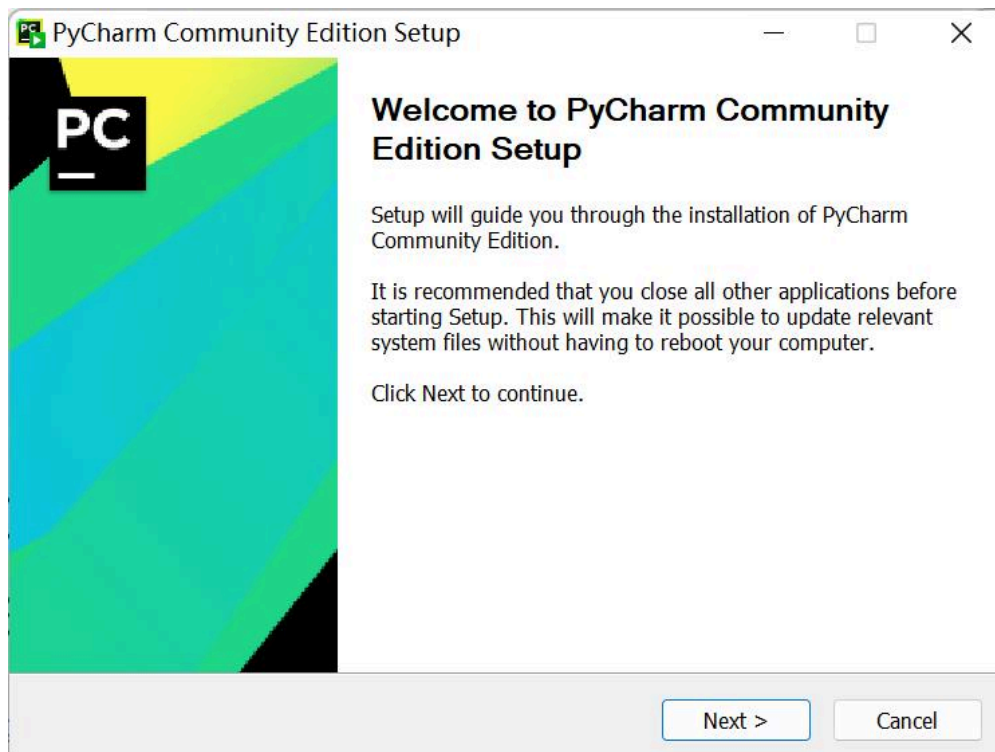
Free, built on open-source



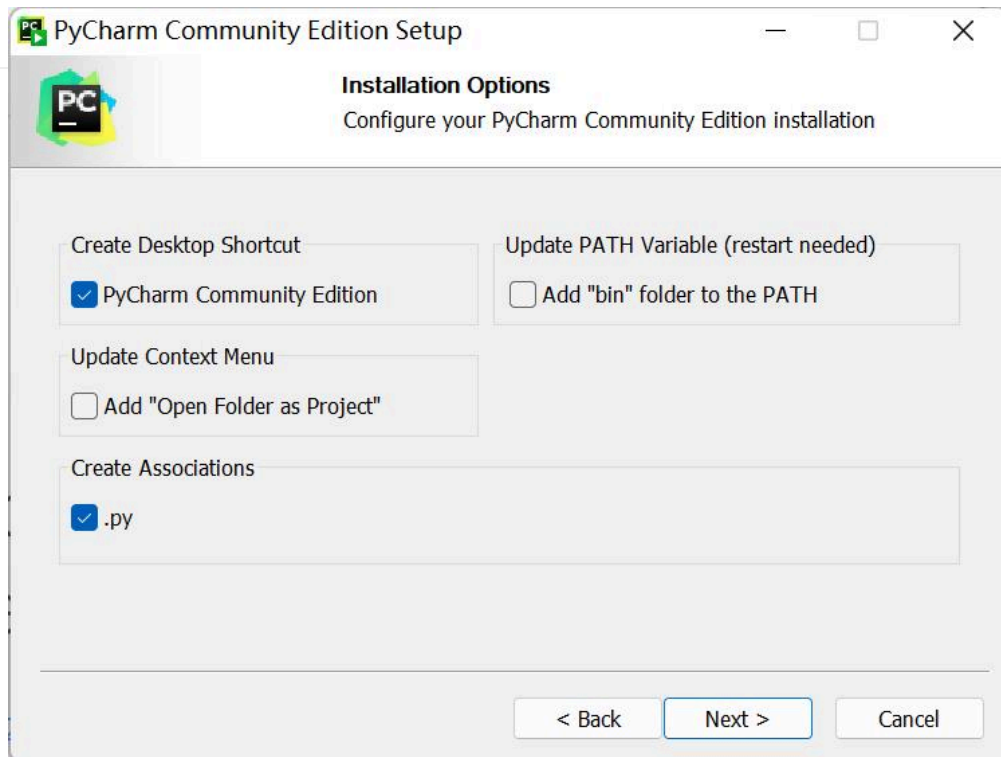
Get the Toolbox App to download PyCharm and its future updates with ease

It is recommended to install the free version.

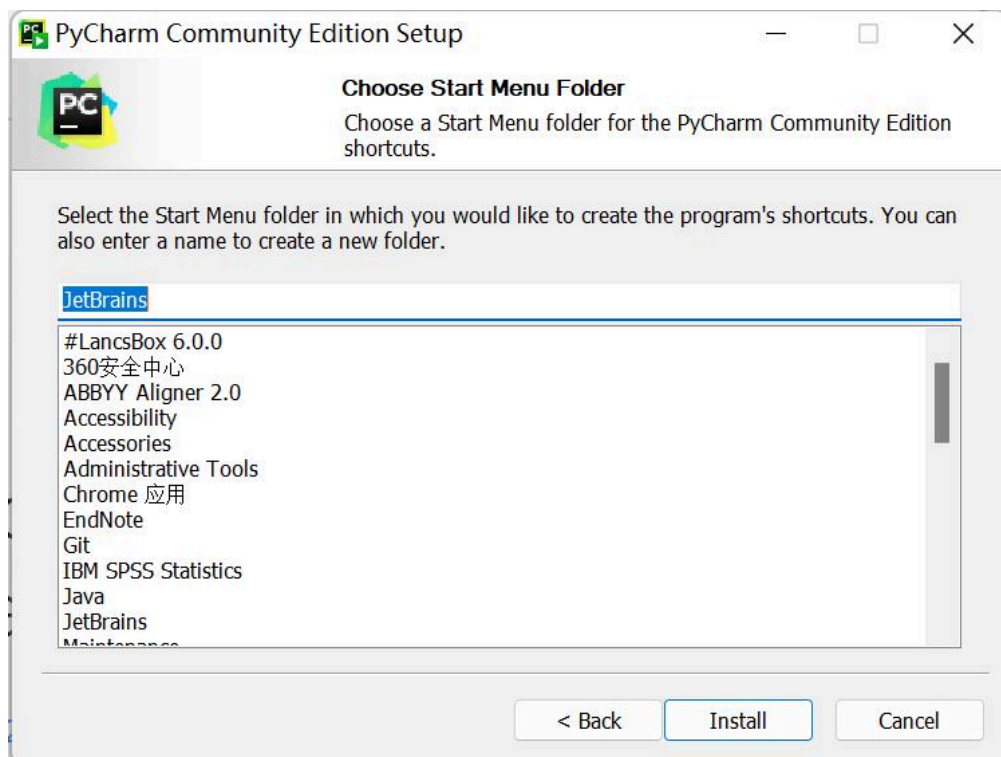
- Click on **Next** :



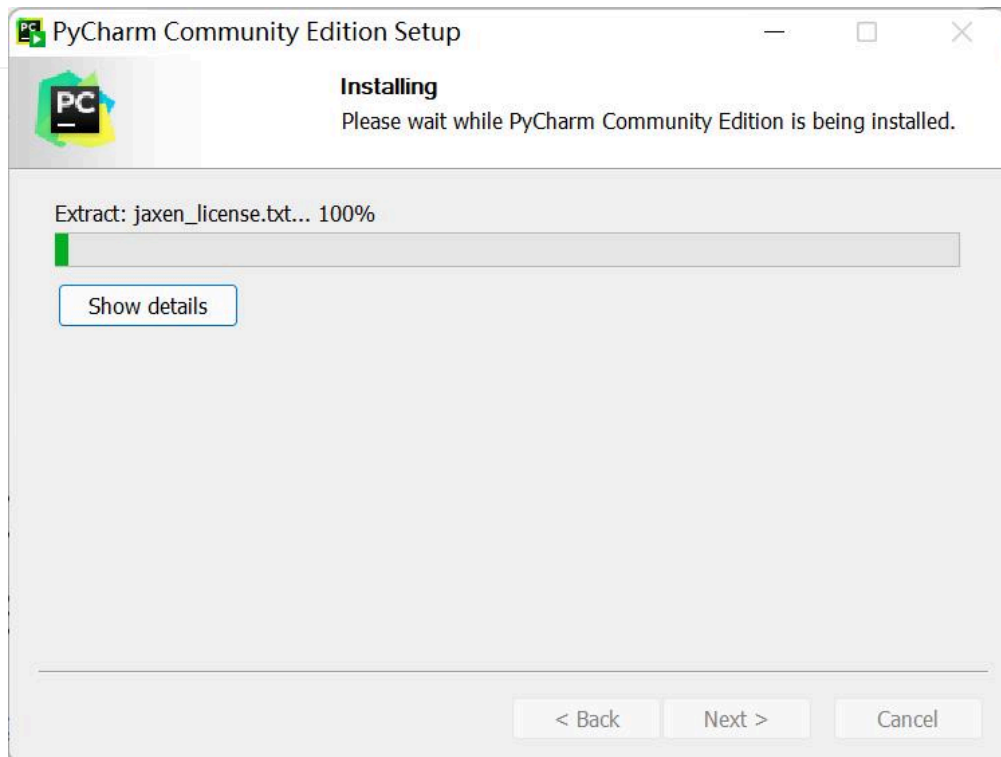
- Select options according to your needs and then select **Next** :



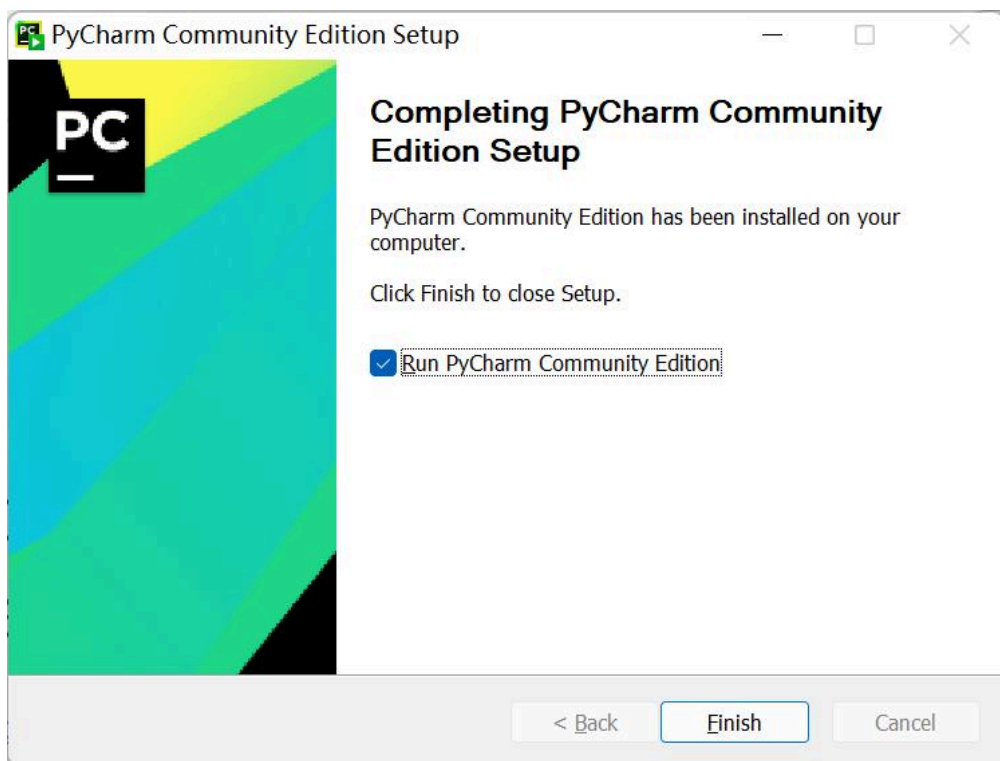
- Tap **Install** :



- Installing:

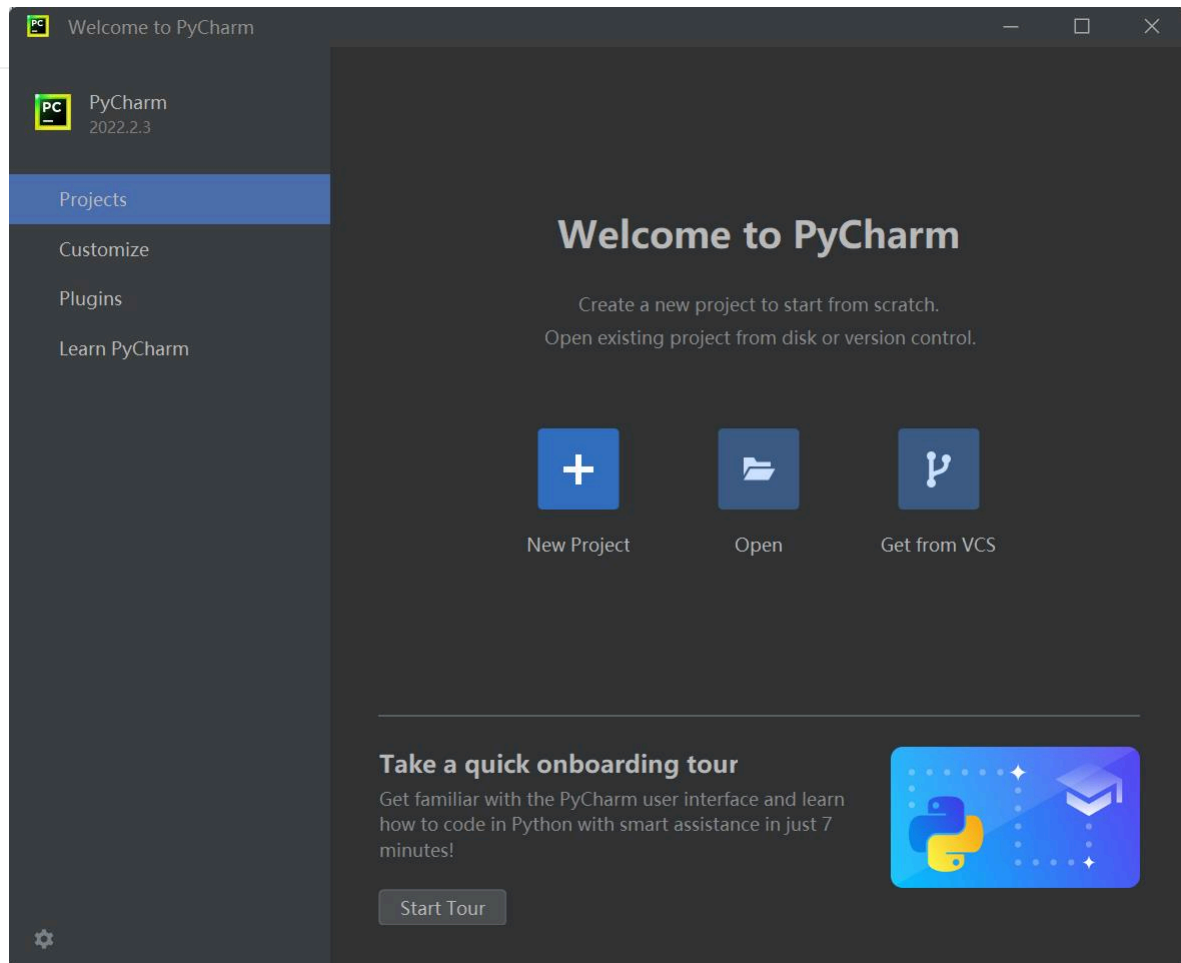


- Tap **Finish**

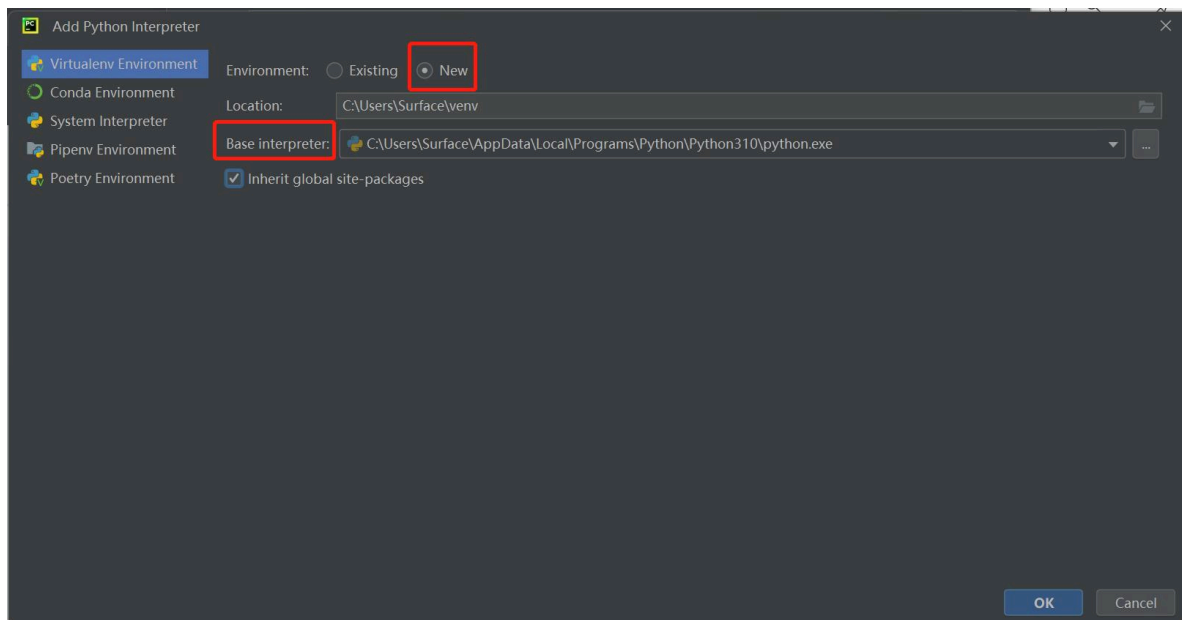
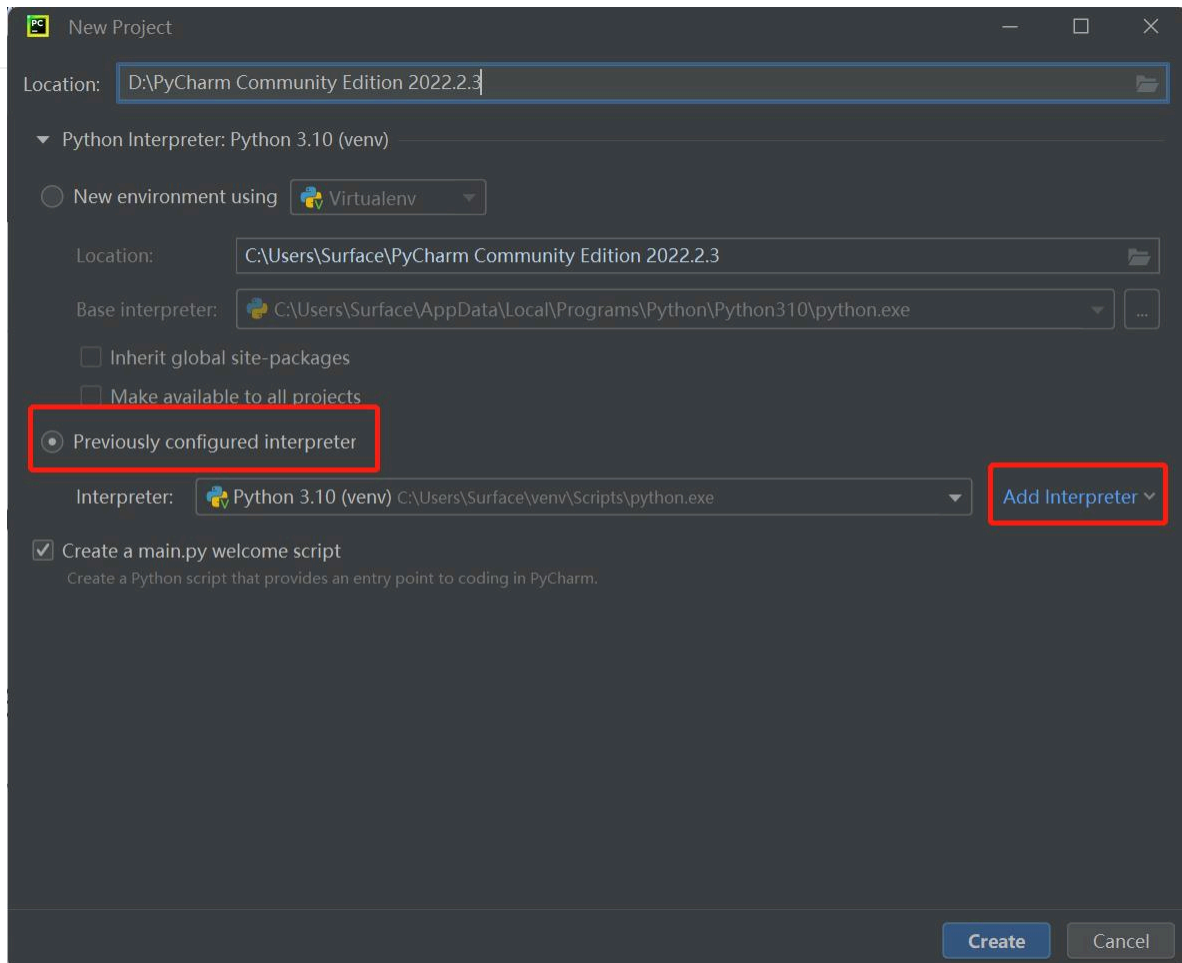


2.2 Create a new project

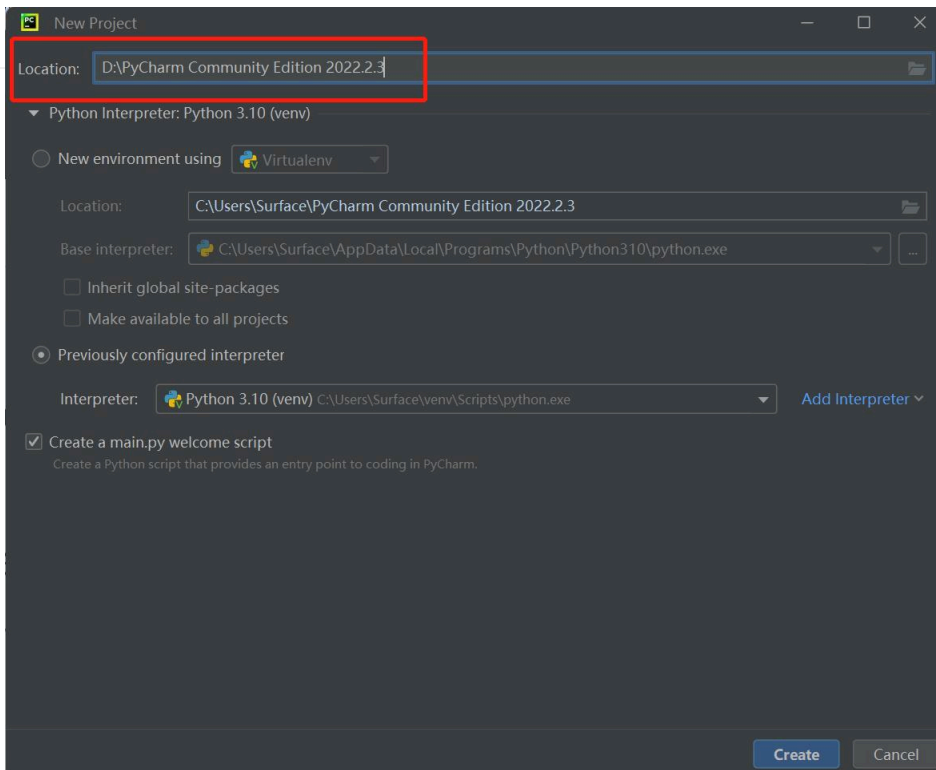
- Click **+New Project** :



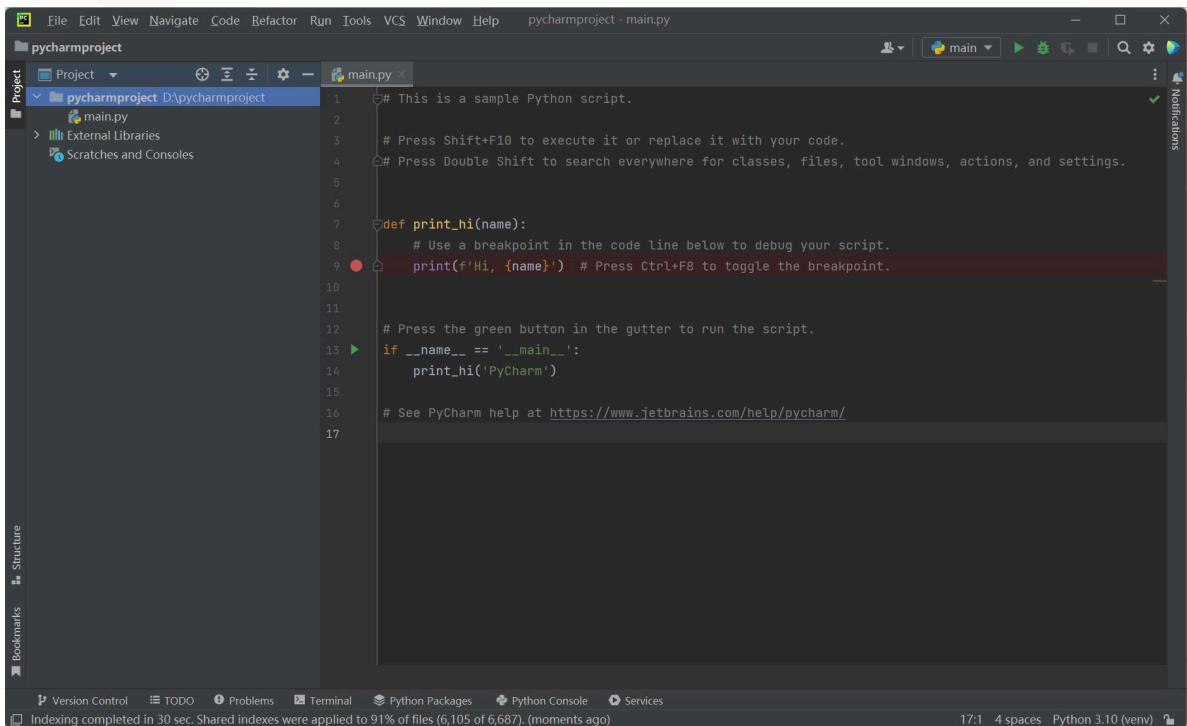
- The `Interpreter` is used to interpret python programs. Select `Add Interpreter` -> `New` to add base interpreter.



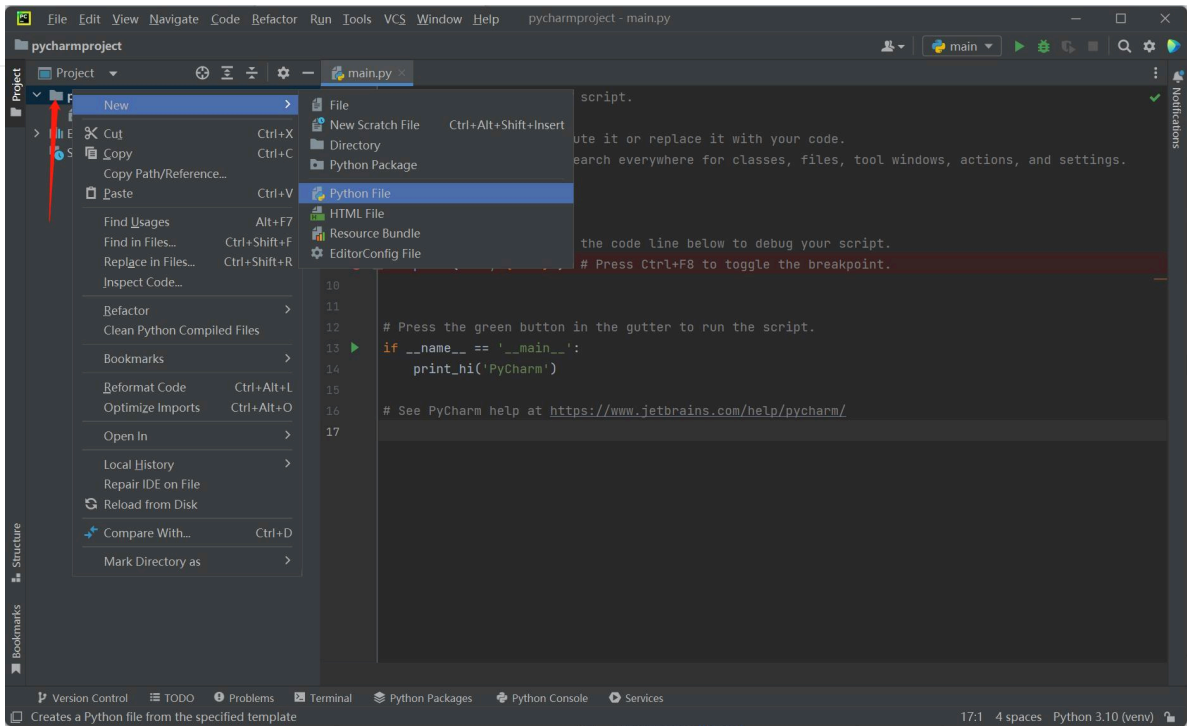
- **Location** refers to the place where to save python file. Choose a file to put your programs.



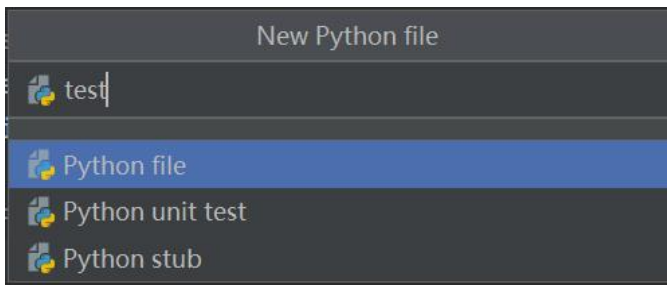
- Click on **Create** and a sample appears:



- Right click on the selection that the red arrow points to, and create a new python file.



- Type name for the new file.



3 Preparations

- Firmware burning. Firmware serves as a driver for systems to control robots.
 - **M5Stack version** Make sure to burn `minirobot` for **Basic** at the bottom. And then choose `Transponder` function (to receive instructions from Basic), Press `Press A`. `Atom: OK` means connect successfully. Refer to [MyStudio](#) for more information about firmware burning.
 - **Pi \ jetsonnano version** `AtomMain` for **Atom** at the top is factory burnt.
- pymycobot installation. Type `pip install pymycobot --upgrade --user` via terminal (Win+R) `cmd` command.

```
pip install pymycobot --upgrade --user
```



- Source code installation. Open a terminal (Win+R, input `cmd`), and type the command below to install.

```
git clone https://github.com/elephantrobotics/pymycobot.git <your-path>
#Fill in your installation address in <your-path>, do not choose the current default path.

cd <your-path>/pymycobot
#Go to the pymycobot folder of the downloaded package.

#Run one of the following commands according to your python version.
# Install
python2 setup.py install
# or
python3 setup.py install
```

- Update pymycobot

```
pip install pymycobot --upgrade
```

4 Import of pymycobot

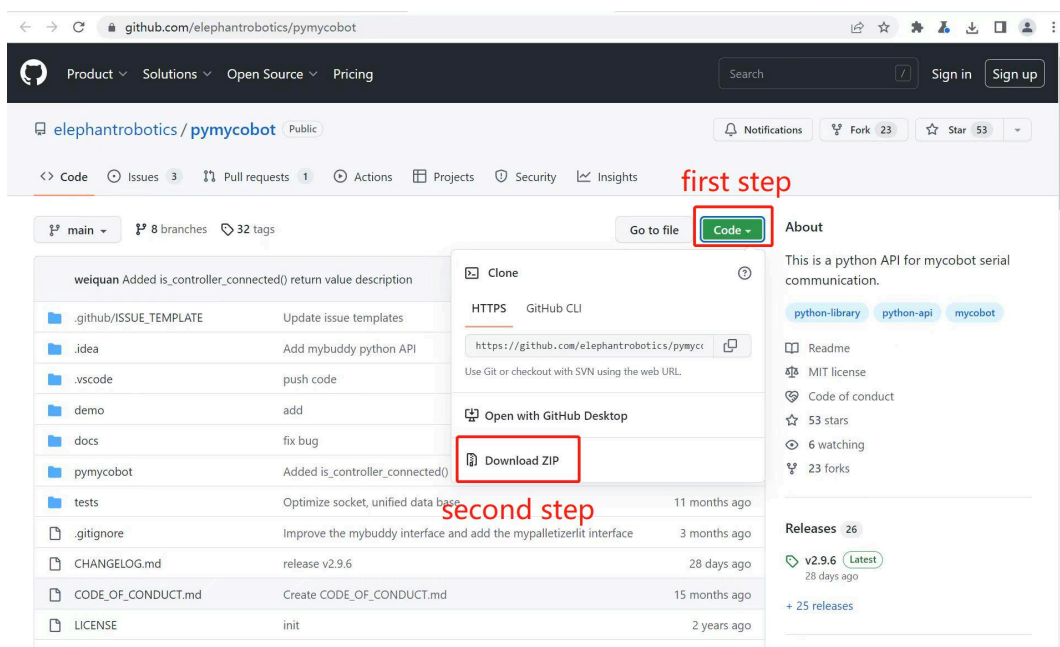
This part takes MyPalletizer 260-M5 as an example to introduce how to control a robot via python.

- Import pymycobot library for MyPalletizer :

```
from pymycobot.mypalletizer import MyPalletizer
```

Notice:

1. If no red wavy line appears below the codes, pymycobot is successfully installed.
2. if a red wavy line appears, got to the address <https://github.com/elephantrobotics/pymycobot> to download pymycobot manually and put it into python library.



5 Simple Demo

Create a new Python file, and type the following codes to set the color of RGB light panel. The codes are suitable to myCobot 280-M5, myCobot 320-M5 and myPalletizer 260.

Notice: The baud rates are different according to types of devices. Refer to [calculator device manager](#) to check the corresponding number.

- **Codes for MyCobot:**

```
# demo.py
from pymycobot.mycobot import MyCobot

from pymycobot import PI_PORT, PI_BAUD      # When using the Raspberry Pi version of mycobot, you can refer to these two v
import time

#The above codes are required to be written, which means importing the project package

# MyCobot class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux:  "/dev/ttyUSB0"
#       or  "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is:  115200
#
#   Example:
#       mycobot-M5:
#           linux:
#               mc = MyCobot("/dev/ttyUSB0", 115200)
#           or mc = MyCobot("/dev/ttyAMA0", 115200)
#           windows:
#               mc = MyCobot("COM3", 115200)
#       mycobot-raspi:
#           mc = MyCobot(PI_PORT, PI_BAUD)
#
# Initiate MyCobot
# Create object code here for windows version
mc = MyCobot("COM3", 115200)

i = 7
#loop 7 times
while i > 0:
    mc.set_color(0,0,255) #blue light on
    time.sleep(2)        #wait for 2 seconds
    mc.set_color(255,0,0) #red light on
    time.sleep(2)        #wait for 2 seconds
    mc.set_color(0,255,0) #green light on
    time.sleep(2)        #wait for 2 seconds
    i -= 1
```

Run the example file:

```
python3 demo.py
```

- **Codes for MyPalletizer:**

```
from pymycobot.mypalletizer import MyPalletizer
import time

##The above codes are required to be written at the beginning, which means importing the project package

#initiate MyPalletizer
mc = MyPalletizer("COM3", 115200)

# MyPalletizer initiation requires two parameters:
# The first is the serial port string, such as:
# linux: "/dev/ttyUSB0"
# or "/dev/ttyAMA0"
# windows: "COM3"
# The second is the baud rate::
# M5 version is: 115200
#
# Example:
# MyPalletizer-M5:
# linux:
# mc = MyPalletizer("/dev/ttyUSB0", 115200)
# or mc = MyPalletizer("/dev/ttyAMA0", 115200)
# windows:
# mc = MyPalletizer("COM3", 115200)

i = 7
#loops 7 times
while i > 0:
    mc.set_color(0,0,255) #blue light on
    time.sleep(2) #wait for 2 seconds
    mc.set_color(255,0,0) #red light on
    time.sleep(2) #wait for 2 seconds
    mc.set_color(0,255,0) #green light on
    time.sleep(2) #wait for 2 seconds
    i -= 1
```

The blue, red, and green lights on the top of the robot flash 7 times continuously at an interval of 2 seconds.

MechArm 270

[toc]

Python API usage instructions

API (Application Programming Interface), also known as Application Programming Interface functions, are predefined functions. When using the following function interfaces, please import our API library at the beginning by entering the following code, otherwise it will not run successfully:

```
# Example
from pymycobot import MechArm270

# M5 version
mc = MechArm270('COM3', 115200)

# PI version
# mc = MechArm270('/dev/ttyAMA0', 1000000)

# Gets the current angle of all joints
angles = mc.get_angles()
print(angles)

# Set 1 joint to move to 40 and speed to 20
mc.send_angle(1, 40, 20)
```

1. System Status

1.1 get_system_version()

- **function:** get system version
- **Return value:** system version

1.2 get_basic_version()

- **function:** Get basic firmware version for M5 version
- **Return value:** float firmware version

1.3 get_error_information()

- **function:** Obtaining robot error information
- **Return value:**
 - 0: No error message.
 - 1 ~ 6: The corresponding joint exceeds the limit position.
 - 16 ~ 19: Collision protection.
 - 32: Kinematics inverse solution has no solution.
 - 33 ~ 34: Linear motion has no adjacent solution.

1.4 clear_error_information()

- **function:** Clear robot error message
- **Return value:**
 - 1 : completed

2. Overall Status

2.1 power_on()

- **function:** atom open communication (default open)
- **Return value:**
 - 1 : completed

2.2 power_off()

- **function:** Power off of the robotic arm
- **Return value:**
 - 1 : completed

2.3 is_power_on()

- **function:** judge whether robot arms is powered on or not
- **Return value:**
 - 1 : power on
 - 0 : power off
 - -1 : error

2.4 release_all_servos(data=None)

- **function:** release all robot arms
 - **Attentions:** After the joint is disabled, it needs to be enabled to control within 1 second
- **Parameters:** `data` (optional): The way to relax the joints. The default is damping mode, and if the 'data' parameter is provided, it can be specified as non damping mode (1- Undamping).
- **Return value:**
 - 1 : completed

2.5 focus_servo(servo_id)

- **function:** Power on designated servo
- **Parameters:**
 - `servo_id`: int, 1-6
- **Return value:**
 - 1 : completed

2.6 is_controller_connected()

- **function:** Whether connected with Atom

- **Return value:**

- 1 : succeed
- 0 : failed
- -1 : error data

2.7 read_next_error()

- **function:** Robot Error Detection

- **Return value:** list len 6

- 0 : No abnormality
- 1 : Communication disconnected
- 2 : Unstable communication
- 3 : Servo abnormality

2.8 get_fresh_mode()

- **function:** Query sports mode

- **Return value:**

- 0 : Interpolation mode
- 1 : Refresh mode

2.9 set_fresh_mode()

- **function:** Set command refresh mode

- **Parameters:**

- 1 : Always execute the latest command first.
- 0 : Execute instructions sequentially in the form of a queue.

- **Return value:**

- 1 : completed

2.10 focus_all servos()

- **Function:** All servos are powered on

- **Return value:**

- 1 : complete

2.11 set_vision_mode()

- **Function:** Set the vision tracking mode, limit the attitude flip of send_coords in refresh mode. (Applicable only to vision tracking function)

- **Parameter:**

- 1 : open
- 0 : close

- **Return value:**

- 1 : complete

3.MDI Mode and Operation

3.1 `get_angles()`

- **function:** get the degree of all joints
- **Return value:** `list` a float list of all degree

3.2 `send_angle(id, degree, speed)`

- **function:** send one degree of joint to robot arm
- **Parameters:**
 - `id` : Joint id(`genre.Angle`), range int 1-6
 - `degree` : degree value(`float`)
 - `speed` : the speed and range of the robotic arm's movement 1~100

Joint Id	Range
1	-165 ~ 165
2	-90 ~ 90
3	-180 ~ 6
4	-165 ~ 165
5	-115 ~ 115
6	-175 ~ 175

- **Return value:**
 - `1` : completed

3.3 `send_angles(angles, speed)`

- **function:** Send all angles to all joints of the robotic arm
- **Parameters:**
 - `angles` : a list of degree value(`List[float]`), length 6
 - `speed` : (`int`) 1 ~ 100
- **Return value:**
 - `1` : completed

3.4 `get_coords()`

- **function:** Obtain robot arm coordinates from a base based coordinate system
- **Return value:** a float list of coord:[x, y, z, rx, ry, rz]

3.5 `send_coord(id, coord, speed)`

- **function:** send one coord to robot arm
- **Parameters:**
 - `id` :send one coord to robot arm, 1-6 corresponds to [x, y, z, rx, ry, rz]
 - `coord` : coord value(`float`)

- o `speed : ( int ) 1-100`

Coord Id	Range
x	-272 ~ 272
y	-272 ~ 272
z	-36 ~ 408.9
rx	-180 ~ 180
ry	-180 ~ 180
rz	-180 ~ 180

- **Return value:**
 - o `1` : completed

3.6 `send_coords(coords, speed, mode)`

- **function::** Send overall coordinates and posture to move the head of the robotic arm from its original point to your specified point
- **Parameters:**
 - o `coords :` a list of coords value `[x,y,z,rx,ry,rz]` ,length6
 - o `speed (int) :` 1 ~ 100
 - o `mode: (int)` 0 - angular, 1 - linear
- **Return value:**
 - o `1` : completed

3.7 `pause()`

- **function:** Control the instruction to pause the core and stop all movement instructions
- **Return value:**
 - o `1` - stopped
 - o `0` - not stop
 - o `-1` - error

3.8 `sync_send_angles(angles, speed, timeout=15)`

- **function:** Send the angle in synchronous state and return when the target point is reached
- **Parameters:**
 - o `angles :` a list of degree value(`List[float]`), length 6
 - o `speed : ( int ) 1 ~ 100`
 - o `timeout :` default 15 s
- **Return value:**
 - o `1` : completed

3.9 `sync_send_coords(coords, speed, mode=0, timeout=15)`

- **function:** Send the coord in synchronous state and return when the target point is reached
- **Parameters:**

- `coords` : a list of coord value(`List[float]`), length 6
- `speed` : (`int`) 1 ~ 100
- `mode` : (`int`) 0 - angular (default) , 1 - linear
- `timeout` : default 15 s
- **Return value:**
 - 1 : completed

3.10 `get_angles_coords()`

- **function:** Get joint angles and coordinates
- **Return value:** A list with a length of 12. The first six digits are angle information, and the last six digits are coordinate information.

3.11 `is_paused()`

- **function:** Check if the program has paused the move command
- **Return value:**
 - 1 - paused
 - 0 - not paused
 - -1 - error

3.12 `resume()`

- **function:** resume the robot movement and complete the previous command
- **Return value:**
 - 1 : completed

3.13 `stop()`

- **function:** stop all movements of robot
- **Return value:**
 - 1 - stopped
 - 0 - not stop
 - -1 - error

3.14 `is_in_position(data, flag)`

- **function** : judge whether in the position.
- **Parameters:**
 - `data`: Provide a set of data that can be angles or coordinate values. If the input angle length range is 6, and if the input coordinate value length range is 6
 - `flag` data type (value range 0 or 1)
 - 0 : angle
 - 1 : coord
- **Return value:**
 - 1 - true
 - 0 - false
 - -1 - error

3.15 is_moving()

- **function:** judge whether the robot is moving
- **Return value:**
 - `1` moving
 - `0` not moving
 - `-1` error

3.16 angles_to_coords(angles)

- **Function :** Convert angles to coordinates.
- **Parameters:**
 - `angles` : `list` List of floating points for all angles.
- **Return value:** `list` List of floating points for all coordinates.

3.17 solve_inv_kinematics(target_coords, current_angles)

- **Function :** Convert coordinates to angles.
- **Parameters:**
 - `target_coords` : `list` List of floating points for all coordinates.
 - `current_angles` : `list` List of floating points for all angles, current angles of the robot
- **Return value:** `list` List of floating points for all angles.

4. JOG Mode and Operation

4.1 jog_angle(joint_id, direction, speed)

- **function:** jog control angle
- **Parameters:**
 - `joint_id` : Represents the joints of the robotic arm, represented by joint IDs ranging from 1 to 6
 - `direction(int)` : To control the direction of movement of the robotic arm, input `0` as negative value movement and input `1` as positive value movement
 - `speed` : 1 ~ 100
- **Return value:**
 - `1` : completed

4.2 jog_coord(coord_id, direction, speed)

- **function:** jog control coord.
- **Parameters:**
 - `coord_id` : (`int`) Coordinate range of the robotic arm: 1~6
 - `direction` : (`int`) To control the direction of machine arm movement, `0` - negative value movement, `1` - positive value movement
 - `speed` : 1 ~ 100
- **Return value:**
 - `1` : completed

4.3 jog_rpy(end_direction, direction, speed)

- **function:** Rotate the end around a fixed axis in the base coordinate system

- **Parameters:**

- `end_direction : ( int )` Roll, Pitch, Yaw (1-3)
- `direction : ( int )` To control the direction of machine arm movement, `1` - forward rotation, `0` - reverse rotation
- `speed : ( int )` 1 ~ 100

- **Return value:**

- `1` : completed

4.4 `jog_increment_angle(joint_id, increment, speed)`

- **Function:** Angle stepping, single joint angle increment control

- **Parameter:**

- `joint_id` : 1-6
- `increment` : Incremental movement based on the current position angle
- `speed` : 1~100

- **Return value:**

- `1` : Completed

4.5 `jog_increment_coord(id, increment, speed)`

- **Function:** Coordinate stepping, single coordinate increment control

- **Parameter:**

- `id` : Coordinate axis 1-6
- `increment` : Incremental movement based on the current position coordinate
- `speed` : 1~100

- **Return value:**

- `1` : Completed

4.6 `set_encoder(joint_id, encoder, speed)`

- **function:** Set a single joint rotation to the specified potential value

- **Parameters**

- `joint_id : ( int )` 1-6
- `encoder` : 0 ~ 4096
- `speed` : 1 ~ 100

4.7 `get_encoder(joint_id)`

- **function:** Set a single joint rotation to the specified potential value

- **Parameters**

- `joint_id : ( int )` 1-6

- **Return value:** `( int )` Joint potential value

4.8 `set_encoders(encoders, speed)`

- **function:** Set the six joints of the manipulator to execute synchronously to the specified position.

- **Parameters**

- `joint_id` : (`int`) 1-6
- `encoder` : 0 ~ 4096
- `speed` : 1 ~ 100

- **Return value:**

- `1` : completed

4.9 `get_encoders()`

- **function:** Get the six joints of the manipulator.
- **Return value:** (`list`) the list of encoders

5. Running status and Settings

5.1 `get_joint_min_angle(joint_id)`

- **function:** Gets the minimum movement angle of the specified joint
- **Parameters:**
 - `joint_id` : Enter joint ID (range 1-6)
- **Return value:** `float` Angle value

5.2 `get_joint_max_angle(joint_id)`

- **function:** Gets the maximum movement angle of the specified joint
- **Parameters:**
 - `joint_id` : Enter joint ID (range 1-6)
- **Return value:** `float` Angle value

5.3 `set_joint_min(id, angle)`

- **function:** Set minimum joint angle limit
- **Parameters:**
 - `id` : Enter joint ID (range 1-6)
 - `angle` : Refer to the limit information of the corresponding joint in the [send_angle\(\)](#) interface, which must not be less than the minimum value
- **Return value:**
 - `1` : completed

5.6 `set_joint_max(id, angle)`

- **function:** Set maximum joint angle limit
- **Parameters:**
 - `id` : Enter joint ID (range 1-6)
 - `angle` : Refer to the limit information of the corresponding joint in the [send_angle\(\)](#) interface, which must not be greater than the maximum value
- **Return value:**
 - `1` : completed

6. Joint motor control

6.1 `is_servo_enable(servo_id)`

- **function:** Detecting joint connection status
- **Parameters:** `servo_id` 1-6
- **Return value:**
 - `1` : Connection successful
 - `0` : not connected
 - `-1` : error

6.2 `is_all_servo_enable()`

- **function:** Detect the status of all joint connections
- **Return value:**
 - `1` : Connection successful
 - `0` : not connected
 - `-1` : error

6.3 `set_servo_calibration(servo_id)`

- **function:** The current position of the calibration joint actuator is the angle zero point
- **Parameters:**
 - `servo_id` : 1 - 6
- **Return value:**
 - `1` : completed

6.4 `release_servo(servo_id)`

- **function:** Set the specified joint torque output to turn off
- **Parameters:**
 - `servo_id` : 1 ~ 6
- **Return value:**
 - `1` : release successful
 - `0` : release failed
 - `-1` : error

6.5 `focus_servo(servo_id)`

- **function:** Set the specified joint torque output to turn on
- **Parameters:** `servo_id` : 1 ~ 6
- **Return value:**
 - `1` : focus successful
 - `0` : focus failed
 - `-1` : error

6.6 `set_servo_data(servo_id, data_id, value, mode=None)`

- **function:** Set the data parameters of the specified address of the steering gear
 - **Parameters:**
-

- `servo_id : ( int )` joint id 1 - 6
- `data_id : ( int )` Data address
- `value : ( int )` 0 - 4096
- `mode` : 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.
- **Return value:**
 - `1` : completed

6.7 `get_servo_data(servo_id, data_id, mode=None)`

- **function:** Read the data parameter of the specified address of the steering gear.
- **Parameters:**
 - `servo_id : ( int )` joint id 1 - 6
 - `data_id : ( int )` Data address
 - `mode` : 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.
- **Return value:** 0 ~ 4096

6.8 `joint_brake(joint_id)`

- **function:** Make it stop when the joint is in motion, and the buffer distance is positively related to the existing speed
- **Parameters:**
 - `joint_id : ( int )` joint id 1 - 6
- **Return value:**
 - `1` : completed

7. Servo state value

7.1 `get_servo_speeds()`

- **function:** Get the movement speed of all joints
- **Return value:** A list unit step/s

7.2 `get_servo_voltages()`

- **function:** Get joint voltages
- **Return value:** A list volts < 24 V

7.3 `get_servo_status()`

- **function:** Get the movement status of all joints
- **Return value:** A list,[voltage, sensor, temperature, current, angle, overload], a value of `0` means no error, a value of `1` indicates an error

7.4 `get_servo_temps()`

- **function:** Get joint temperature
- **Return value:** A list unit °C

8. Robotic arm end IO control

8.1 `set_color(r, g, b)`

- **function:** Set the color of the end light of the robotic arm
- **Parameters:**
 - `r (int)` : 0 ~ 255
 - `g (int)` : 0 ~ 255
 - `b (int)` : 0 ~ 255
- **Return value:**
 - `1` : completed

8.2 `set_digital_output(pin_no, pin_signal)`

- **function:** set IO statue
- **Parameters**
 - `pin_no (int)`: Pin number
 - `pin_signal (int)`: 0 / 1
- **Return value:**
 - `1` : completed

8.3 `get_digital_input(pin_no)`

- **function:** read IO statue
- **Parameters:** `pin_no (int)`
- **Return value:** signal

8.4 `set_pin_mode(pin_no, pin_mode)`

- **function:** Set the state mode of the specified pin in atom.
- **Parameters**
 - `pin_no (int)`: Pin number
 - `pin_mode (int)`: 0 - input, 1 - output, 2 - input_pullup
- **Return value:**
 - `1` : completed

9. Robotic arm end gripper control

9.1 `set_gripper_state(flag, speed, _type_1=None, is_torque=None)`

- **function:** Adaptive gripper enable
 - **Parameters:**
 - `flag (int)` : 0 - open 1 - close, 254 - release
 - `speed (int)` : 1 ~ 100
 - `_type_1 (int)` :
-

- 1 : Adaptive gripper (default state is 1)
-
- 2 : A nimble hand with 5 fingers
 - 3 : Parallel gripper
 - 4 : Flexible gripper
 - is_torque (int): Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-device Atom firmware version ≥ 1.3**)
 - 0 : Non-force-controlled gripper
 - 1 : Force-controlled gripper
 - **Return value:**
 - 1 : completed

9.2 set_gripper_value(gripper_value, speed, gripper_type=None, is_torque=None)

- **function:** Set the gripper value
- **Parameters:**
 - gripper_value (int) : 0 ~ 100
 - speed (int) : 1 ~ 100
 - gripper_type (int) :
 - 1 : Adaptive gripper (default state is 1)
 - 3 : Parallel gripper
 - 4 : Flexible gripper
 - is_torque (int): Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-device Atom firmware version ≥ 1.3**)
 - 0 : Non-force-controlled gripper
 - 1 : Force-controlled gripper
- **Return value:**
 - 1 : completed

9.3 set_gripper_calibration()

- **function:** Set the current position of the gripper to zero

is_gripper_moving()

- **function:** Judge whether the gripper is moving or not
- **Return value:**
 - 0 : not moving
 - 1 : is moving
 - -1 : error data

9.4 `get_gripper_value()`

- **function:** Get the value of gripper
- **Parameters:**
 - `gripper_type` : (int) default 1
 - 1: Adaptive gripper
 - 3: Parallel gripper
 - 4: Flexible gripper
- **Return value:** gripper value (int)

9.5 `set_pwm_output(channel, frequency, pin_val)`

- **function:** PWM control
- **Parameters:**
 - `channel` : (int): IO number.
 - `frequency` : (int): clock frequency
 - `pin_val` : (int) Duty cycle 0 ~ 256; 128 means 50%
- **Return value:**
 - 1 : completed

9.6 `set HTS_gripper_torque(torque)`

- **function:** Set new adaptive gripper torque
- **Parameters:**
 - `torque` : (int): 150 ~ 980
- **Return value:**
 - 0 : Set failed
 - 1 : Set successful

9.7 `get HTS_gripper_torque()`

- **function:** Get gripper torque
- **Return value:** (int) 150 ~ 980

9.8 `get_gripper_protect_current()`

- **function:** Get the gripper protection current
- **Return value:** (int) 1 ~ 500

9.9 `set_gripper_protect_current(current)`

- **function:** Set the gripper protection current
- **Parameters:**
 - `current` : (int): 1 ~ 500
- **Return value:**
 - 1 : completed

9.10 `init_gripper()`

- **function:** Initialize gripper

10. Set bottom IO input/output status

10.1 `set_basic_output(pin_no, pin_signal)`

- **function:** Set Base IO Output
- **Parameters:**
 - `pin_no` (`int`) Pin port number
 - `pin_signal` (`int`): 0 - low. 1 - high
- **Return value:**
 - `1` : completed

10.2 `get_basic_input(pin_no)`

- **function:** Read base IO input
- **Parameters:**
 - `pin_no` (`int`) pin number
- **Return value:** 0 - low. 1 - high

11. WLAN Setting

11.1 `set_ssid_pwd(account, password)`

- **function:** Change connected wifi. (Apply to m5)
- **Parameters:**
 - `account` (`str`) new wifi account
 - `password` (`str`) new wifi password
- **Return value:**
 - `1` : completed

11.2 `get_ssid_pwd()`

- **function:** Get connected wifi account and password. (Apply to m5)
- **Return value:** (account, password)

11.3 `set_server_port(port)`

- **function:** Change the connection port of the server
- **Parameters:**
 - `port` (`int`) The new connection port of the server.
- **Return value:**
 - `1` : completed

12. TOF

12.1 `get_tof_distance()`

- **function:** Get the detected distance (Requires external distance detector)
- **Return value:** (int) The unit is mm.

13. Communication mode

13.1 `set_transponder_mode(mode)`

- **function:** Set basic communication mode
- **Parameters:**
 - `mode` : 0 - Turn off transparent transmission, 1 - Open transparent transmission
- **Return value:**
 - `1` : completed

13.2 `get_transponder_mode()`

- **function:** Get basic communication mode
- **Parameters:**
- **Return value:**
 - `1` : Open transparent
 - `0` : Turn off transparent transmission

14. Cartesian space coordinate parameter setting

14.1 `set_tool_reference(coords)`

- **function:** Set tool coordinate system.
- **Parameters:**
 - `coords` : (`list`) [x, y, z, rx, ry, rz].
- **Return value:**
 - `1` : completed

14.2 `get_tool_reference(coords)`

- **function:** Get tool coordinate system.
- **Return value:**
 - `coords` : (`list`) [x, y, z, rx, ry, rz]

14.3 `set_world_reference(coords)`

- **function:** Set world coordinate system.
- **Parameters:**
 - `coords` : (`list`) [x, y, z, rx, ry, rz].
- **Return value:**
 - `1` : completed

14.4 `get_world_reference()`

- **function:** Get world coordinate system.
- **Return value:** `list` [x, y, z, rx, ry, rz].

14.5 `set_reference_frame(rftype)`

- **function:** Set base coordinate system.
- **Parameters:**

- `rftype` : 0 - base 1 - tool.

- **Return value:**

- `1` : completed

14.6 `get_reference_frame()`

- **function:** Get base coordinate system.
- **Return value:** (`list`) [x, y, z, rx, ry, rz].

14.7 `set_movement_type(move_type)`

- **function:** Set movement type.
- **Parameters:**
 - `move_type` : 1 - moveI, 0 - moveJ.
- **Return value:**
 - `1` : completed

14.8 `get_movement_type()`

- **function:** Get movement type.
- **Return value:**
 - `1` - moveI
 - `0` - moveJ

14.9 `set_end_type(end)`

- **function:** Set end coordinate system
- **Parameters:**
 - `end (int)` : `0` - flange, `1` - tool
- **Return value:**
 - `1` : completed

14.10 `get_end_type()`

- **function:** Obtain the end coordinate system
- **Return value:**
 - `0` - flange
 - `1` - tool

15. Raspberry pi -- GPIO

15.1 `gpio_init()`

- **function:** Init GPIO module, and set BCM mode.

15.2 `gpio_output(pin, v)`

- **function:** Set GPIO port output value.
- **Parameters**
 - `pin ( int )` Pin number.

o `v ( int )`: 0 / 1

16. utils (module)

This module supports some helper methods. Use the code entered at the beginning of the file to import the module:

```
from pymycobot import utils
```

16.1 `utils.get_port_list()`

- **Function:** Get a list of all current serial port numbers
- **Return value:** Serial port list (`list`)

16.2 `utils.detect_port_of_basic()`

- **Function:** Return the first detected serial port number of M5 Basic. (Only one serial port number will be returned)
- **Return value:** Return the detected port number. If no serial port number is detected, it will return: `None`

MechArm 270 Socket

Note: raspberryPi version Only supports python3 The robotic arm that uses this class of premise has a server and has been turned on.

Use TCP/IP to control the robotic arm

1. Client

```
# demo
from pymycobot import MechArmSocket
# Port 9000 is used by default
mc = MechArmSocket("192.168.10.10",9000)

res = mc.get_angles()
print(res)

mc.send_angles([0,0,0,0,0,0],20)
...
```

2. Server

Server file is in the `demo folder` ,For details, please check the [Server_270.py](#) file in the demo folder

3. socket control

Note: Most of the methods are the same as the class `MechArm270`, only the new methods are listed here.

3.1 `set_gpio_mode(mode)`

- **function:** Set pin coding method.
- **Parameters**
 - `mode ( str )` "BCM" or "BOARD".

3.2 `set_gpio_out(pin_no, mode)`

- **function:** Set the pin as input or output.
- **Parameters**
 - `pin_no ( int )` pin id.
 - `mode ( str )` "in" or "out"

3.3 `set_gpio_output(pin_no, state)`

- **function:** Set the pin to high or low level.
- **Parameters**
 - `pin_no ( int )` pin id.
 - `state ( int )` 0 or 1

3.4 `get_gpio_in(pin_no)`

- **function:** Get pin level status.
- **Parameters**
 - `pin_no ( int )` pin id.
- **Return value:** `0` is low level `1` is high level

Joint Control

For a serial multi-joint robot, the control of the joint space is to control the variables of each joint so as to make each joint reaches a target position at a certain speed.

Notice: When setting the angle, the values corresponding to different manipulators are different. Refer to the parameter introduction section for more information.

myPalletizer 260

Simple Demo

```
from pymycobot.mypalletizer import MyPalletizer260
import time
# import the project package

# Initiate a MyPalletizer260 object, M5 version
mc = MyPalletizer260("COM3",115200)

# PI version
# mc = MyPalletizer260("/dev/ttyAMA0",1000000)

# By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, spe
mc.send_angles([0, 0, 0, 0,], 50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# Move joint 1 to the 50 position
mc.send_angle(1,20,50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)
# set variable "num"
num = 2
# set the number of loops
while num > 0:
    mc.send_angle(2,20,50)
    time.sleep(2)
    mc.send_angle(2,(-20),50)
    time.sleep(2)
    num -= 1

# make robot arms reach the specified position
mc.send_angles([-0.87, 41.66, -12.13, -0.17], 50)
# Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```

mechArm 270

Single-Joint Control

send_angle(id, degree, speed)

- **Function:** to sends a specified single joint motion to a specified angle.
- **Parameters:**
 - `id` : to stand for the joints of a robot arm. represented by numbers 1-6.
 - `degree` : means the angle of a joint.
 - `speed` : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** 1

set_encoder(joint_id, encoder, speed)

- **Function:** to sends a specified single joint motion to a specified potential value.
- **Parameters:**
 - `joint_id` : to stand for the joints of a robot arm. represented by numbers 1-6.
 - `encoder` : means the potential value of the robot arm, ranging from 0 - 4096.
 - `speed` : means the movement speed of the robot arm, ranging from 1 to 100.
- **Return value:** 1

Multi-Joint Control

get_angles()

- **Function:** to get the angels of all joints.
- **Return Value:** `List` : a list of floating point values which represent the angles of all joints

send_angles(degrees, speed)

- **Function:** to send all angles to all joints.
- **Parameters:**
 - `degrees` : (List [float]) contains the angles of all joints.the length is 6; The representation method is [20, 20,20, 20, 20, 20]
 - `speed` : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** 1

set_encoders(encoders, sp)

- **Function:** Send potential values to all joints of the robotic arm.
- **Parameters:**
 - `encoder` : means the potential of the robot arm, ranging from 0 - 4096, length is 6. The way to represent: [2048, 2048, 2048, 2048, 2048, 2048].
 - `sp` : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** 1

sync_send_angles(degrees, speed, timeout=15)

- **Function:** to send an angle synchronously; return when reaching a target point.
 - **Parameters:**
 - `degrees` : A list of angle values of each joint `List[float]` .
 - `speed` : (`int`) means the movement speed of the robot arm, ranging from 0 to 100.
 - `timeout` : The default time is 15s.
-

- **Return value:** 1

get_radians()

- **Function:** to get the radian of all joints.
- **Return value:** `list` : a list containing radian values of all joints

send_radians(radians, speed)

- **Function:** to send radian values to all joints.
- **Parameters:**
 - `radians : list` : a list containing radian values of all joints, ranging from -5 to 5.
- **Return value:** 1

Simple Demo

```

from pymycobot.mecharm270 import MechArm270

import time

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux:  "/dev/ttyUSB0"
#       or      "/dev/ttyACM0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#   Example:
#       mecharm-M5:
#           linux:
#               mc = MechArm270("/dev/ttyUSB0", 115200)
#           or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#               mc = MechArm270("COM3", 115200)
#       mecharm-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initiate a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)

# PI version
# mc = MechArm270("/dev/ttyAMA0", 1000000)

#By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, 0, 0]
mc.send_angles([0, 0, 0, 0, 0, 0], 50)

# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2.5)

# Move joint 1 to the 90 position
mc.send_angle(1, 90, 50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# The following code can make the robotic arm swing left and right
# set the number of loops
while num > 0:

    # Move joint 2 to the 50 position
    mc.send_angle(2, 50, 50)

    # Set the waiting time to ensure that the robotic arm has reached the specified position

```

```
time.sleep(1.5)

# Move joint 2 to the -50 position
mc.send_angle(2, -50, 50)

# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(1.5)

num -= 1

#Make the robotic arm retract. You can manually swing the robotic arm, and then use the get_angles() function to get the c
# use this function to let the robotic arm reach the position you want.
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)

# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2.5)

# Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```

myBuddy

single joint control

send_angle(id, joint, angle, speed)

- **Function** Send one degree of joint to robot arm.
- **Parameters**
 - **id** – 1/2/3 (L/R/W)
 - **joint** – 1 ~ 6
 - **angle** – int
 - **speed** – 1 ~ 100
- **Returns**
 - None

get_angle(id, joint_id)

- **Function** Get the angle of a single joint
- **Parameters**
 - **id** (*int*) – 1/2/3 (L/R/W).
 - **joint_id** (*int*) – 1 - 7 (7 is gripper)

set_encoder(id, joint_id, encoder, speed)

- **Function** Set a single joint rotation to the specified potential value.

- **Parameters**

-
- **id** – 1/2/3 (L/R/W).
 - **joint_id** – 1 - 6.
 - **encoder** – The value of the set encoder.

- **Returns**

- None

multi-joint control

get_angles(id)

- **Function** Get the degree of all joints.

- **Parameters**

id – 1/2 (L/R)

- **Returns**

A float list of all degree.

- **Return type**

list

send_angles(id, degrees, speed)

- **Function** Send all angles to the robotic arm

- **Parameters**

- **id** – 1/2 (L/R).
- **degrees** – [angle_list] len 6
- **speed** – 1 - 100

set_encoders(id, encoders, speed)

- **Function** Set the six joints of the manipulator to execute synchronously to the specified position.

- **Parameters**

- **id** – 1/2 (L/R).
- **encoders** – A encoder list, length 6.
- **speed** – speed 1 ~ 100

get_radians(id)

- **Function** Get the radians of all joints

- **Parameters**

id – 1/2 (L/R)

- **Returns**

A list of float radians [radian1, ...]

- **Return type**

list

send_radians(id, radians, speed)

- **Function** Send the radians of all joints to robot arm
- **Parameters**
 - **id** – 1/2 (L/R).
 - **radians** – a list of radian values(List[float]), length 6
 - **speed** – (int)1 ~ 100

Simple Demo

```
from pymycobot.mybuddy import MyBuddy
import time
mc = MyBuddy("/dev/ttyACM0", 115200)

# Send angles to the six joints of the left arm
mc.send_angles(1, [0, 0, 0, 0, 0, 0], 50)
time.sleep(3)

# Send the angle to the first joint of the left arm
mc.send_angle(1, 1, 90, 50)
time.sleep(2)

# Get the joint angle of the left arm
angles = mc.get_angles(1)
print("left angles: ",angles)

# Relax all joints of the left arm. Before running this command, please support the left arm with your hand to prevent it
mc.release_all_servos(1)
```

myArm

Simple Demo

```

from pymycobot.myarm import MyArm
from pymycobot.genre import Angle
import time

# import the project package

# Initiate a MyArm object
mc = MyArm("/dev/ttyAMA0", 115200)

# By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, 0,
mc.send_angles([0, 0, 0, 0, 0, 0], 50)

# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# Move joint 1 to the 90 position
mc.send_angle(1, 90, 50)

# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# set variable "num"
num = 2

# set the number of loops
while num > 0:
    mc.send_angle(2, 20, 50)
    time.sleep(2)
    mc.send_angle(2, -20, 50)
    time.sleep(2)
    num -= 1

# make robot arms reach the specified position
mc.send_angles([-5.25, -30, -20, -12, -10, -10, 10], 50)

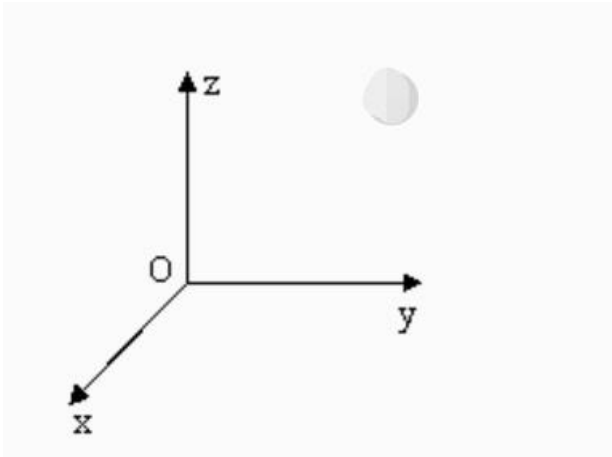
# Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()

```

Coordinate Control

It is mainly used to make intelligent route planning to move the robot arms from one position to another specified position. The coordinate is $[x, y, z, rx, ry, rz]$. $[x, y, z]$ represents the position of the robot arm head in space (the coordinate system is [cartesian coordinate system](#)). $[rx, ry, rz]$ represents the posture of such head at this point (the coordinate system is euler coordinates). The above simple explanation helps you to use functions better.

Note: When setting the coordinates, different series of manipulators have different joint structures. For the same set of coordinates, different series of manipulators will show different postures.



myPalletizer 260

Simple Demo

```
from pymycobot.mypalletizer import MyPalletizer260
import time
# import the project package

# Initiate a MyPalletizer260 object, M5 version
mc = MyPalletizer260("COM3", 115200)

# PI version
# mc = MyPalletizer260("/dev/ttyAMA0", 1000000)

# # Get the current coordinates and pose of the head
coords = mc.get_coords()
print(coords)

#Plan the route at random, let the head reach the coordinates of [57.0, -107.4, 316.3] in an non-linear manner at the sp
mc.send_coords([187.8, 42.1, 183.3, -159.6], 80, 0)
# wait for 2 seconds
time.sleep(2)

# Plan the route at random, let the head reach the coordinates of [207.9, 47, 49.3,-159.69] in an non-linear manner at t
mc.send_coords([207.9, 47, 49.3,-159.69], 80, 0)
# wait for 2 seconds
time.sleep(2)

#To change only the x-coordinate of the head, set the x-coordinate of the head to 20. Let it plan the route at random an
mc.send_coord(1, 20, 50)
```

mechArm 270

Single-Parameter Coordinate

send_coord(id,coord,speed)

- **Function:** to send a single coordinate value to the robot arm to make it move.
- **Parameter:**
 - **id** : 1-6 represents the coordinates of the robotic arm. For example, you can fill in 1 for X-axis, 2 for Y-axis, and so on.
 - **coord** : Input the coordinate value you want.
 - **speed** : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return Value:** 1

Multiple parameter coordinates

get_coords()

- **Function:** to obtain the current coordinate and posture.
- **Return Value:** `list` : a list containing coordinates and postures.
 - Six axes: The length is 6, and they are `[x, y, z, rx, ry, rz]` in order.

send_coords(coords, speed, mode)

- **Function:** to send the overall coordinates and postures to move the robot arm head from the original point to the point you have specified.
- **Parameters:**
 - `coords` :
 - Six axes: The length of the coordinate value of `[x, y, z, rx, ry, rz]` is 6.
 - Four axes: The length of the coordinate value of `[x,y,z,rx]` is 4.
 - `speed` : means the movement speed of the robot arm, ranging from 0 to 100.
 - `mode` : (`int`): The value is limited to 0 and 1.
 - 0 means that the movement path of the robot arm head is non-linear, i.e. the movement route is randomly planned just to make sure that the head moves to a specified point with a specified posture.
 - 1 means that the movement path of the robot arm head is linear, i.e. the movement route is intelligently planned just to make sure that the head moves to a specified point with a specified posture in a linear manner.
- **Return Value:** None

set_tool_reference(coords)

- **Function:** Set Tool coordinate system.
- **Parameters:**
 - `coords` : The coordinate value of `[x, y, z, rx, ry, rz]` has a length of 6, x, y, z ranging from - 280 to 280, and rx, ry, rz ranging from - 314 to 314
- **Return Value:** None

get_tool_reference()

- **Function:** Get Tool coordinate system.
- **Return Value:** Returns a coordinate list with a length of 6

get_world_reference()

- **Function:** Get World coordinate system.
- **Return Value:** Returns a coordinate list with a length of 6

set_world_reference(coords)

- **Function:** Set World coordinate system.
- **Parameters:**
 - `coords` : The coordinate value of `[x, y, z, rx, ry, rz]` has a length of 6, x, y, z ranging from - 280 to 280, and rx, ry, rz ranging from - 314 to 314
- **Return Value:** None

set_reference_frame(rftype)

- **Function:** Set Base coordinate system.
- **Parameters:**
 - `rftype` : 0 - Base coordinate system(default), 1 - World coordinate system
- **Return Value:** None

get_reference_frame()

- **Function:** Get Base coordinate system.

- **Return Value:** 0 - Base coordinate system, 1 - World coordinate system, -1 - error

set_end_type(end)

- **Function:** Set end coordinate system.
- **Parameters:**
 - `end` : 0 - flange(default), 1 - tool
- **Return Value:** None

get_end_type()

- **Function:** Get end coordinate system
- **Return Value:** 0 - flange(default), 1 - tool, -1 - error

Simple Demo

```

from pymycobot.mecharm270 import MechArm270
import time

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux:  "/dev/ttyUSB0"
#       or      "/dev/ttyACM0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#   Example:
#       MechArm270-M5:
#           linux:
#               mc = MechArm270("/dev/ttyUSB0", 115200)
#           or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#               mc = MechArm270("COM3", 115200)
#       MechArm270-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initialize a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)

# PI version
# mc = MechArm270("/dev/ttyAMA0", 1000000)
# Get the current coordinates and pose of the head
coords = mc.get_coords()
print(coords)

# Intelligently plan the route, let the head reach the coordinates of [152, -9.5, 220.8] in a linear manner, and maintain
mc.send_coords([152, -9.5, 220.8, 143.29, 2, 88], 80)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

# Intelligently plan the route, let the head reach the coordinates of [124, -9.5, 232] in a linear way, and maintain the a
mc.send_coords([124, -9.5, 232, 143.29, 2, 88], 80)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

# To change only the x-coordinate of the head, set the x-coordinate of the head to -40. Let it plan the route intelligently
mc.send_coord(1, -40, 70)

```

myBuddy

One-parameter coordinates

send_coord(id, coord, data, speed)

- **Function** Send a single coordinate to the robotic arm
- **Parameters**
 - **id** – 1/2/3 (L/R/W).
 - **coord** – 1 ~ 6 (x/y/z/rx/ry/rz)
 - **data** – Coordinate value
 - **speed** – 0 ~ 100

get_coord(id, joint_id)

- **Function** Read a single coordinate parameter
- **Parameters**
 - **id** (*int*) – 1/2/3 (L/R/W).
 - **joint_id** (*int*) – 1 - 7 (7 is gripper)

Multiparameter Coordinates

send_coords(id, coords, speed, mode)

- **Function** Send all coords to robot arm.
- **Parameters**
 - **id** – 1/2 (L/R).
 - **coords** – a list of coords value(List[float]), length 6, [x(mm), y, z, rx(angle), ry, rz]
 - **speed** – (int) 0 ~ 100
 - **mode** – (int) 0 - moveJ, 1 - moveL, 2 - moveC

Simple Demo

```
from pymycobot.mybuddy import MyBuddy
import time

mc = MyBuddy("/dev/ttyACM0", 115200)

# Get the coordinates and posture of the current head of the left arm
coords = mc.get_coords(1)
print(coords)

# Intelligently plan the route, let the head reach the coordinates of [57.0, -107.4, 316.3] in a linear manner, and maintain
mc.send_coords(1, [57.0, -107.4, 316.3, -93.81, -12.71, -163.49], 80, 1)

time.sleep(1.5)

# Intelligently plan the route, let the head reach the coordinates of [-13.7, -107.5, 223.9] in a linear manner, and maintain
mc.send_coords(1, [-13.7, -107.5, 223.9, 165.52, -75.41, -73.52], 80, 1)

time.sleep(1.5)

# To change only the x-coordinate of the head of the left arm, set the x-coordinate of the head to -40. Let it plan the route
mc.send_coord(1, 1, -40, 70)
```

myArm

Simple Demo

```
#from pymycobot.myarm import MyArm
#from pymycobot.genre import Coord
#import time

# Initialize a MyArm object
# Create object code here for windows version
mc = MyArm("/dev/ttyAMA0", 115200)

# Get the current coordinates and pose of the head
coords = mc.get_coords()
print(coords)

# Intelligently plan the route, let the head reach the coordinates of [57.0, -107.4, 316.3, -93.81, -12.71, -163.49] in
mc.send_coords([57.0, -107.4, 316.3, -93.81, -12.71, -163.49], 80,0)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

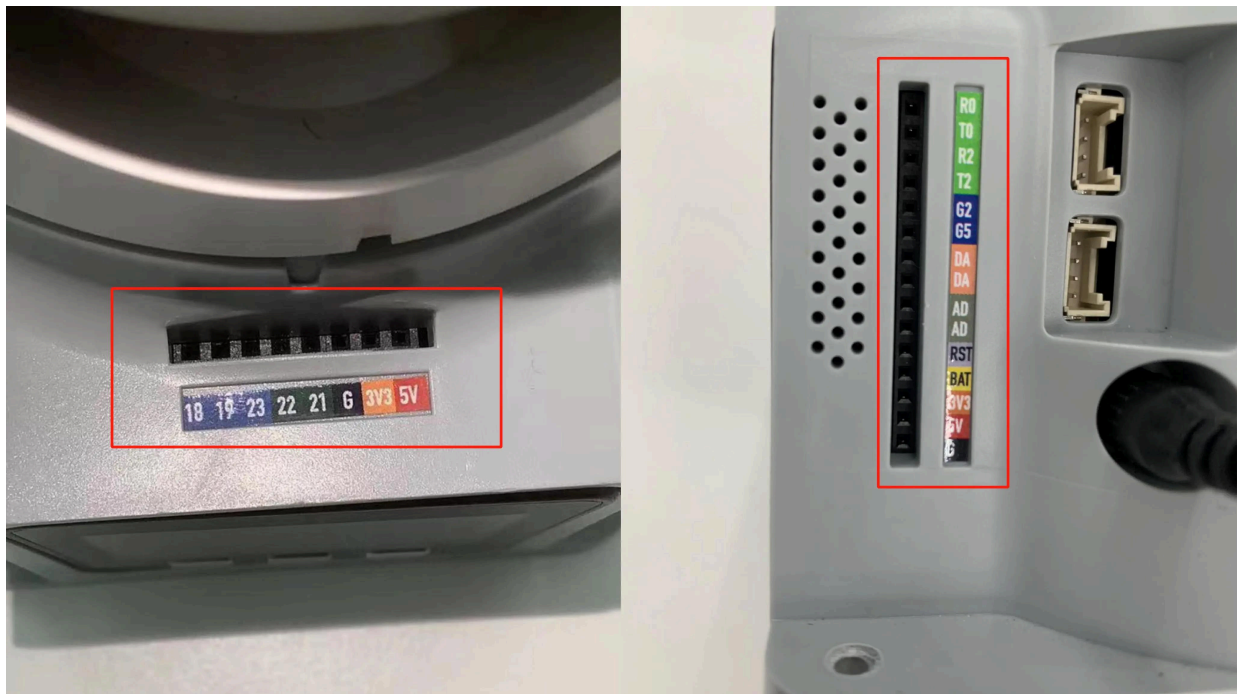
# Intelligently plan the route, let the head reach the coordinates of [-13.7, -107.5, 223.9, 165.52, -75.41, -73.52] in
mc.send_coords([-13.7, -107.5, 223.9, 165.52, -75.41, -73.52], 80, 0)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

# To change only the x-coordinate of the head, set the x-coordinate of the head to 20. Let it plan the route intelligent
mc.send_coord(Coord.X.value, 20, 70)
```

IO control

IO is the input and output of data. There are multiple pins on the Basic and Atom of our robot arm. The input and output modes can be set through the following function interface.



myPalletizer 260

Simple Demo

- 260-M5 version:

```
from pymycobot.mypalletizer260 import MyPalletizer260
import time

#Enter the above code to import the packages required by the project

# initiate MyPalletizer260, M5 version
mc = MyPalletizer260("COM3", 115200)

for count in range(5):
    # set a loop
    mc.set_basic_output(2,0)
    # Let the basic2 position enter the working state
    mc.set_basic_output(5,0)
    # Let the basic5 position enter the working state
    time.sleep(2)
    #等待两秒
    mc.set_basic_output(2,1)
    #Let the basic2 position stop working
    mc.set_basic_output(5,1)
    #Let the basic5 position stop working
```

- 260-PI version:

```
from pymycobot.mypalletizer260 import MyPalletizer260
import time

#Enter the above code to import the packages required by the project

# initiate MyPalletizer260, PI version
mc = MyPalletizer260("/dev/ttyAMA0", 1000000)

# initialization
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
# open suction pump
GPIO.output(20, 0)
GPIO.output(21, 0)
# wait 2 seconds
time.sleep(2)
# Turn off the suction pump
GPIO.output(20, 1)
GPIO.output(21, 1)
```

mechArm 270

Basic IO

get_basic_input(pin_no)

- **Function:** to obtain the working state of the bottom pin number

- **Parameters:** `pin_no` : represents the specific pin number at the bottom of the robot.
- **Return Value:** `pin_signal ( int )` When the returned value is 0, it means running in the working state; when it is 1, it means the stop state.

set_basic_output(pin_no, pin_signal)

- **Function:** to set the working state of the bottom pin number.
- **Parameters:**
 - `pin_no ( int )`. Only the numerical part of the numbers marked at the bottom of the equipment is taken.
 - `pin_signal ( int )`: Inputting 0 means setting to the running state, which inputting 1 means setting to the stop state
- **Return Value:** 1

get_tof_distance()

- **Function:** to obtain a detected distance (An external distance detector is required).
- **Return value:** The detected distance value (in mm).

Atom IO

set_pin_mode(pin_no, pin_mode)

- **Function:** to set the state mode of the specified pin in the atom.
- **Parameters:**
 - `pin_no (int)`: represents the specific pin number on the top of the robot.
 - `pin_mode (int)`: limited to 0-2
 - 0 means setting it to the running state;
 - 1 means setting it to the stop state;
 - 2 means setting to the pull-up mode.
- **Return Value:** 1

set_digital_output(pin_no, pin_signal)

- **Function:** to set the working state of the end pin number.
- **Parameters:**
 - `pin_no ( int )`. Only the numerical part of the number marked at the end of the equipment is taken.
 - `pin_signal ( int )`. Inputting 0 means setting to the running state, while inputting 1 means setting to the stop state.
- **Return Value:** 1

get_digital_input(self, pin_no)

- **Function:** to obtain the working state of the end pin number.
- **Parameters:** `pin_no` : represents the specific pin number at the end of the robot.
- **Return Value:** `pin_signal ( int )` When the returned value is 0, it means running in the working state; when it is 1, it means the stop state.

Raspberry Pi——GPIO

For Raspberry Pi version, use the following API.

Type the code at the beginning:

```
from pycobot import MechArm270
import RPi.GPIO as GPIO
```

gpio_init()

- **Function:** Initialize GPIO module, set BCM mode.
- **Return value:** None

set_gpio_mode

- **Function:** Set Raspberry Pi GPIO Pin Mode
- **Parameter**
 - `mode ( str )` Input: "BCM" or "BOARD" to enter the corresponding mode

set_gpio_output(pin_no, state)

- **Function:** Set the pin to high, low.
- **Parameter:**
 - `pin ( int )` pin number.
 - `state` : 0 is set to low level 1 is set to high level (low level of suction pump starts working, high level stops working)
 - **Return Value:** None

get_gpio_in(pin_no)

- **Function:** Get the pin level status.
- **Parameter Description:**
 - `pin_no ( int )` pin number.
- **Return Value:** 0 is low level 1 is high level

Simple Demo

270-M5 version:

```
from pymycobot.mecharm import MechArm270

import time

# Enter the above code to import the packages required by the project

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#       or "/dev/ttyACM0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#   Example:
#       MechArm270-M5:
#           linux:
#               mc = MechArm270("/dev/ttyUSB0", 115200)
#           or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#               mc = MechArm270("COM3", 115200)
#       MechArm270-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initialize a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)

for count in range(5):
    # set a loop
    mc.set_basic_output(2,0)
    # Let the basic2 position enter the working state
    mc.set_basic_output(5,0)
    # Let the basic5 position enter the working state
    time.sleep(2)
    #等待两秒
    mc.set_basic_output(2,1)
    #Let the basic2 position stop working
    mc.set_basic_output(5,1)
    #Let the basic5 position stop working
```

270-Pi version:

```
from pymycobot.mecharm import MechArm270
import RPi.GPIO as GPIO
import time

#Enter the above code to import the packages required by the project

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux:  "/dev/ttyUSB0"
#       or      "/dev/ttyACM0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#   Example:
#       MechArm270-M5:
#           linux:
#               mc = MechArm270("/dev/ttyUSB0", 115200)
#           or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#               mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initialize a MechArm270 object
# Create object code here for Raspberry Pi version
mc = MechArm270("/dev/ttyAMA0", 1000000)

# initialization
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
# open suction pump
GPIO.output(20, 0)
GPIO.output(21, 0)
# wait 2 seconds
time.sleep(2)
# Turn off the suction pump
GPIO.output(20, 1)
GPIO.output(21, 1)
```

myBuddy

Atom IO

set_pin_mode(id, pin_no, pin_mode)

- **Function** Set the state mode of the specified pin in atom.
- **Parameters**

- **id** – 1/2 (L/R)

-
- **pin_no** (*int*) – pin number (1 - 5).

- **pin_mode** (*int*) – 0 - input, 1 - output

set_digital_output(id, pin_no, pin_signal)

- **Function** Set atom IO output level

- **Parameters**

- **id** – 1/2 (L/R)
- **pin_no** (*int*) – 1 - 5
- **pin_signal** (*int*) – 0 / 1

get_digital_input(id, pin_no)

- **Function** singal value

- **Parameters**

- **id** – 1/2 (L/R)
- **pin_no** (*int*) – 1 - 5

set_pwm_output(id, channel, frequency, pin_val)

- **Function** PWM control

- **Parameters**

- **id** – 1/2 (L/R)
- **channel** (*int*) – IO number (1 - 5).
- **frequency** (*int*) – clock frequency (0/1: 0 - 1Mhz 1 - 10Mhz)
- **pin_val** (*int*) – Duty cycle 0 ~ 100: 0 ~ 100%

Raspberry Pi IO

set_gpio_input(pin)

- **Function** Set GPIO input value.

- **Parameters**

pin – (int)pin number.

set_gpio_mode(pin_no, mode)

- **Function** Init GPIO module, and set BCM mode.

- **Parameters**

- **pin_no** – (int)pin number.
- **mode** – 0 - input 1 - output

set_gpio_output(pin, v)

- **Function** Set GPIO output value.

- **Parameters**

- **pin** – (int)pin number.
- **v** – (int) 0 / 1

set_gpio_pwm(pin, baud, dc)

- **Function** Set GPIO PWM value.

- **Parameters**

- **pin** – (int)pin number.
- **baud** – (int) 10 - 1000000
- **dc** – (int) 0 - 100

Simple Demo

```
from pymycobot import MyBuddy
import time
mc = MyBuddy("/dev/ttyACM0")

# 设置树莓派IO 20为输出模式
mc.set_gpio_mode(20, 1)

mc.set_gpio_output(20, 1)
time.sleep(2)
mc.set_gpio_output(20, 0)
```

MyArm

Simple Demo

```
from pycobot.mycobot import MyArm
import time

#Enter the above code to import the packages required by the project

# Initialize a MyArm object
mc = MyArm("/dev/ttyAMA0", 115200)

# initialization
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

# open suction pump
GPIO.output(20, 0)
GPIO.output(21, 0)

# wait 2 seconds
time.sleep(2)

# Turn off the suction pump
GPIO.output(20, 1)
GPIO.output(21, 1)
```

Gripper Control

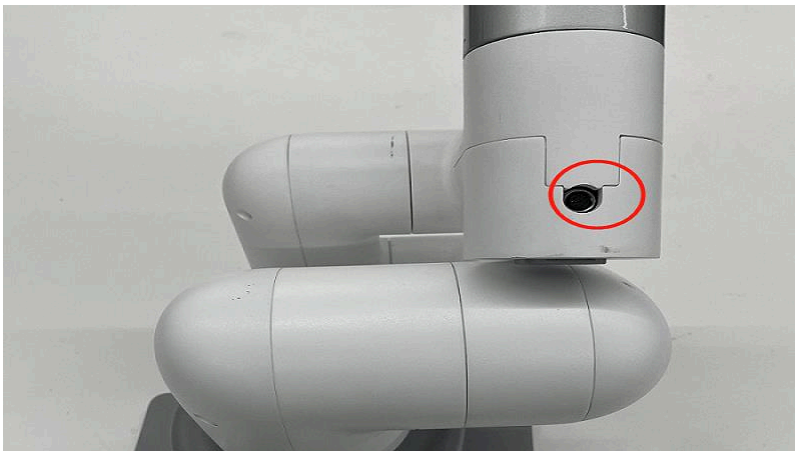
First install and connect the gripper onto the robot arm. Different types of gripper is compatible with different types of robots. Refer to [2.8 Accessories](#) for more information.

Notice:

For MyCobot 280, the adaptive gripper is attached to Atom.



The electric gripper is attached to 495 port.



* MyCobot 280-m5 is not compatible with electric gripper, and MyCobot 320-m5 is only compatible with electric gripper.

myPalletizer 260

Simple Demo

```

from pymycobot.mypalletizer260 import MyPalletizer260
from pymycobot.genre import Angle
import time

#Enter the above code to import the packages required by the project

# initiate MyPalletizer260, M5 version
mc = MyPalletizer260("COM3", 115200)

# PI version
# mc = MyPalletizer260("COM3", 115200)

# let joint2 move to 30 degree at the speed of 50
mc.send_angle(2,30,50)

# waite for 2 seconds
time.sleep(2)

#set a variable num, and then set a loop
num = 5
while num > 0:
    #let gripper open at the speed of 70
    mc.set_gripper_state(0,70)
    # waite for 2 seconds
    time.sleep(2)
    # let gripper close at the speed of 70
    mc.set_gripper_state(1, 70)
    # waite for 2 seconds
    time.sleep(2)
    num -= 1

```

mechArm 270

Controlling Gripper

`is_gripper_moving( )`

- **Function:** Determine whether the gripper is running
- **return value:**
 - o 0 : Indicates that the gripper of the robot arm is not running
 - o 1 : Indicates that the gripper of the robot arm is running
 - o -1 : indicates an error

`set_gripper_state(flag, speed, _type_1=None, is_torque=None)`

- **function:** Adaptive gripper enable
- **Parameters:**
 - o flag (int) : 0 - open 1 - close, 254 - release

- o `speed (int)` : 1 ~ 100

-
- o `_type_1 (int)` :

- `1` : Adaptive gripper (default state is 1)
- `2` : A nimble hand with 5 fingers
- `3` : Parallel gripper
- `4` : Flexible gripper

- o `is_torque (int)`: Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-device Atom firmware version ≥ 1.3**)

- `0` : Non-force-controlled gripper
- `1` : Force-controlled gripper

- **Return value:**

- o `1` : completed

```
set_gripper_value(gripper_value, speed, gripper_type=None, is_torque=None)
```

- **function:** Set the gripper value

- **Parameters:**

- o `gripper_value (int)` : 0 ~ 100
- o `speed (int)` : 1 ~ 100
- o `gripper_type (int)` :

- `1` : Adaptive gripper (default state is 1)
- `3` : Parallel gripper
- `4` : Flexible gripper

- o `is_torque (int)`: Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-device Atom firmware version ≥ 1.3**)

- `0` : Non-force-controlled gripper
- `1` : Force-controlled gripper

- **Return value:**

- o `1` : completed

```
get_gripper_value(gripper_type=None)
```

- **Function:** Get the current position data information of the gripper

- **Parameter Description:**

- o `gripper_type` : Gripper type, the default is adaptive gripper
 - `1` : Adaptive gripper
 - `3` : Parallel jaws
 - `4` : Flexible gripper

- **Return value:** Gripper data information

```
set_electric_gripper(status)
```

- **Function:** Set gripper mode (only for 350)
- **Parameter description:** `status` : 1 means the clamping jaw is closed, 0 means the clamping jaw is open.
- **Return value:** 1

`set_gripper_mode(status)`

- **Function:** Set gripper mode
- **Parameter description:** `status` : 1 transparent transmission mode, 0 I/O mode
- **Return value:** 1

`get_gripper_mode()`

- **Function:** Get gripper status
- **Return value:** `status(int)` : 0 - Transparent transmission mode 1 - I/O mode

`set_HTS_gripper_torque(torque)`

- **Function:** Set adaptive gripper torque
- **Parameter Description:**
 - `torque` : 150 ~ 900
- **Return value:** 0 - Setting failed; 1 - Setting successful

`get_HTS_gripper_torque()`

- **Function:** Get adaptive gripper torque
- **Return value:** 150 ~ 900

`get_gripper_protect_current()`

- **Function:** Get gripper protection current
- **Return value:** 1 ~ 500

`init_gripper()`

- **Function:** Initialize gripper
- **Return value:** 0 - initialization failed; 1 - initialization successful

`set_gripper_protect_current(current)`

- **Function:** Set gripper protection current
- **Parameter Description:**
 - `current` : 1 ~ 500
- **Return value:** 0 - initialization failed; 1 - initialization successful

Simple Demo

```

from pymycobot.mecharm270 import MechArm270
import time

#Enter the above code to import the packages required by the project

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to the position of 2048
    mc.set_encoder(1, 2048)
    time.sleep(2)
    # Set six joint positions and let the robotic arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(3)

    # Let the gripper reach the state of 100 at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Let the gripper reach the state of 0 at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    # Set the state of the gripper to quickly open the gripper at a speed of 70
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # Set the state of the gripper so that it quickly closes the gripper at a speed of 70
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())

if __name__ == "__main__":
    # MechArm270 class initialization requires two parameters:
    # The first is the serial port string, such as:
    # linux: "/dev/ttyUSB0"
    # or "/dev/ttyACM0"
    # windows: "COM3"

```



```

# The second is the baud rate::
#     M5 version is: 115200
#
# Example:
#     MechArm270-M5:
#         linux:
#             mc = MechArm270("/dev/ttyUSB0", 115200)
#             or mc = MechArm270("/dev/ttyACM0", 115200)
#         windows:
#             mc = MechArm270("COM3", 115200)
#     MechArm270-raspi:
#         mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code here for Raspberry Pi version below
mc = MechArm270("/dev/ttyAMA0", 1000000)

# M5 version
# mc = MechArm270("COM3", 115200)

# make it move to zero position
mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)

```

Controlling Gripper

`is_gripper_moving( )`

- **Function:** Determine whether the gripper is running
- **return value:**
 - 0 : Indicates that the gripper of the robot arm is not running
 - 1 : Indicates that the gripper of the robot arm is running
 - -1 : indicates an error

`set_gripper_value(value, speed, gripper_type=None)`

- **Function:** Let the gripper rotate to the specified position at the specified speed
- **Parameter Description:**
 - value : Indicates the position that the clamping jaw wants to reach, the value range is 0~256
 - speed : indicates the speed at which to rotate, the value range is 0~100
 - gripper_type : Gripper type, the default is adaptive gripper
 - 1 : Adaptive gripper
 - 3 : Parallel jaws
 - 4 : Flexible gripper
- **Return value:** None

`get_gripper_value(gripper_type=None)`

- **Function:** Get the current position data information of the gripper
- **Parameter Description:**

- `gripper_type` : Gripper type, the default is adaptive gripper
 - `1` : Adaptive gripper
 - `3` : Parallel jaws
 - `4` : Flexible gripper
- **Return value:** Gripper data information

`set_gripper_state(flag, speed, _type=None)`

- **Function:** Let the gripper enter the specified state at the specified speed
- **Parameter Description:**
 - `flag` : `1` means the clamping jaw is closed, `0` means the clamping jaw is open.
 - `speed` : Indicates how fast to reach the specified state, the value range is `0~100`
 - `_type` : Gripper type, the default is adaptive gripper
 - `1` : Adaptive gripper
 - `2` : Five-fingered dexterity
 - `3` : Parallel jaws
 - `4` : Flexible gripper
- **Return value:** None

`set_electric_gripper(status)`

- **Function:** Set gripper mode (only for 350)
- **Parameter description:** `status` : `1` means the clamping jaw is closed, `0` means the clamping jaw is open.
- **Return value:** None

`set_gripper_mode(status)`

- **Function:** Set gripper mode
- **Parameter description:** `status` : `1` transparent transmission mode, `0` I/O mode
- **Return value:** None

`get_gripper_mode()`

- **Function:** Get gripper status
- **Return value:** `status(int)` : `0` - Transparent transmission mode `1` - I/O mode

`set_HTS_gripper_torque(torque)`

- **Function:** Set adaptive gripper torque
- **Parameter Description:**
 - `torque` : `150 ~ 900`
- **Return value:** `0` - Setting failed; `1` - Setting successful

`get_HTS_gripper_torque()`

- **Function:** Get adaptive gripper torque
- **Return value:** `150 ~ 900`

`get_gripper_protect_current()`

- **Function:** Get gripper protection current
- **Return value:** `1 ~ 500`

`init_gripper()`

- **Function:** Initialize gripper
- **Return value:** `0` - initialization failed; `1` - initialization successful

`set_gripper_protect_current(current)`

- **Function:** Set gripper protection current
 - **Parameter Description:**
 - `current` : 1 ~ 500
 - **Return value:** 0 - initialization failed; 1 - initialization successful
-

Simple Demo

```

from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time

#Enter the above code to import the packages required by the project

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to the position of 2048
    mc.set_encoder(1, 2048)
    time.sleep(2)
    # Set six joint positions and let the robotic arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(3)

    # Let the gripper reach the state of 100 at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Let the gripper reach the state of 0 at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    # Set the state of the gripper to quickly open the gripper at a speed of 70
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # Set the state of the gripper so that it quickly closes the gripper at a speed of 70
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())

if __name__ == "__main__":
    # MyCobot class initialization requires two parameters:
    # The first is the serial port string, such as:
    #     linux: "/dev/ttyAMA0"
    #     or "/dev/ttyAMA0"

```

```
# windows: "COM3"
# The second is the baud rate::
# M5 version is: 115200
#
# Example:
# mycobot-M5:
# linux:
# mc = MyCobot("/dev/ttyAMA0", 1000000)
# or mc = MyCobot("/dev/ttyAMA0", 115200)
# windows:
# mc = MyCobot("COM3", 115200)
# mycobot-raspi:
# mc = MyCobot(PI_PORT, PI_BAUD)
#
# Initialize a MyCobot object
# Create object code here for Raspberry Pi version below
mc = MyCobot(PI_PORT, PI_BAUD)
# make it move to zero position
mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)
```

myBuddy

Controlling Gripper

is_gripper_moving(id)

- **Function** Judge whether the gripper is moving or not

- **Parameters**

id – 1/2 (L/R)

- **Returns**

- 0 - not moving
- 1 - is moving
- -1 - error data

set_gripper_value(id, value, speed)

- **Function** Set gripper value

- **Parameters**

- **id** – 1/2 (L/R)
- **value** (*int*) – 0 ~ 100
- **speed** (*int*) – 0 ~ 100

get_gripper_value(id)

- **Function** Get the value of gripper.

- **Parameters**

id – 1/2 (L/R)

- **Returns**

gripper value (int)

is_gripper_moving(id)

- **Function** Judge whether the gripper is moving or not

- **Parameters**

id – 1/2 (L/R)

- **Returns**

- 0 - not moving
- 1 - is moving
- -1 - error data

myArm

Simple Demo

```

from pymycobot.myarm import MyArm
import time

#Enter the above code to import the packages required by the project

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to the position of 2048
    mc.set_encoder(1, 2048)
    time.sleep(2)
    # Set six joint positions and let the robotic arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024,1024], 20)
    time.sleep(3)

    # Let the gripper reach the state of 100 at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Let the gripper reach the state of 0 at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    # Set the state of the gripper to quickly open the gripper at a speed of 70
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # Set the state of the gripper so that it quickly closes the gripper at a speed of 70
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())

if __name__ == "__main__":

    # Initialize a MyArm object

```

1 myCobot Series 6-Axis Collaborative Robotic Arm

```
mc = MyArm("/dev/ttyAMA0", 115200)
# make it move to zero position
mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)
```


TCP/IP

TCP/IP, or Transmission Control Protocol/Internet Protocol, is one of the most fundamental communicative protocol on Internet, which stipulates the standard and methods of the Internet communication. Users can control robotic arms remotely through connecting with IP address instead of the USB port.

In this chapter, myCobot 280 M5 is used as an example for explanation.

Make sure that M5Stack-basic and Atom are both burnt before using.

MechArm

1.Connection

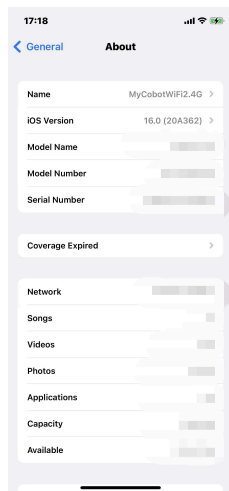
1.1 Create a default network

When using TCP/IP with the myCobot 280 m5 , it connects to a "MyCobotWiFi2.4G" network using the default password "mycobot123".

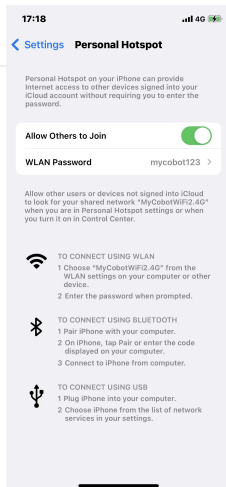
At this point, we can create a hotspot with a mobile phone, rename the hotspot network to "MyCobotWiFi2.4G", and set the password to "mycobot123". After enabling the hotspot, the robotic arm will automatically connect to the mobile phone hotspot using the TCP/IP function. Subsequently, as long as the devices are on the same local area network, they can communicate with each other over the network.

Similarly, you can set up a router by configuring the network name and password. Once the router is set up, and the robotic arm's TCP/IP function is enabled, it will connect to the router.

One important point to note is that the myCobot 280 m5 only supports the 2.4 GHz network band and does not support the 5 GHz network band. The following example uses a mobile phone hotspot.



1 myCobot Series 6-Axis Collaborative Robotic Arm



1.2 Enabled the TCP/IP function

As shown in the figure, the robotic arm connects by pressing the button and selecting Transponder->WLAN Server. If the connection is successful, it will display the IP address and port number. If the connection fails, please check if the network name and password are set correctly.





If the screen shows WIFI Connected, IP and Port, it means that robotic arm is successfully connected with WIFI.

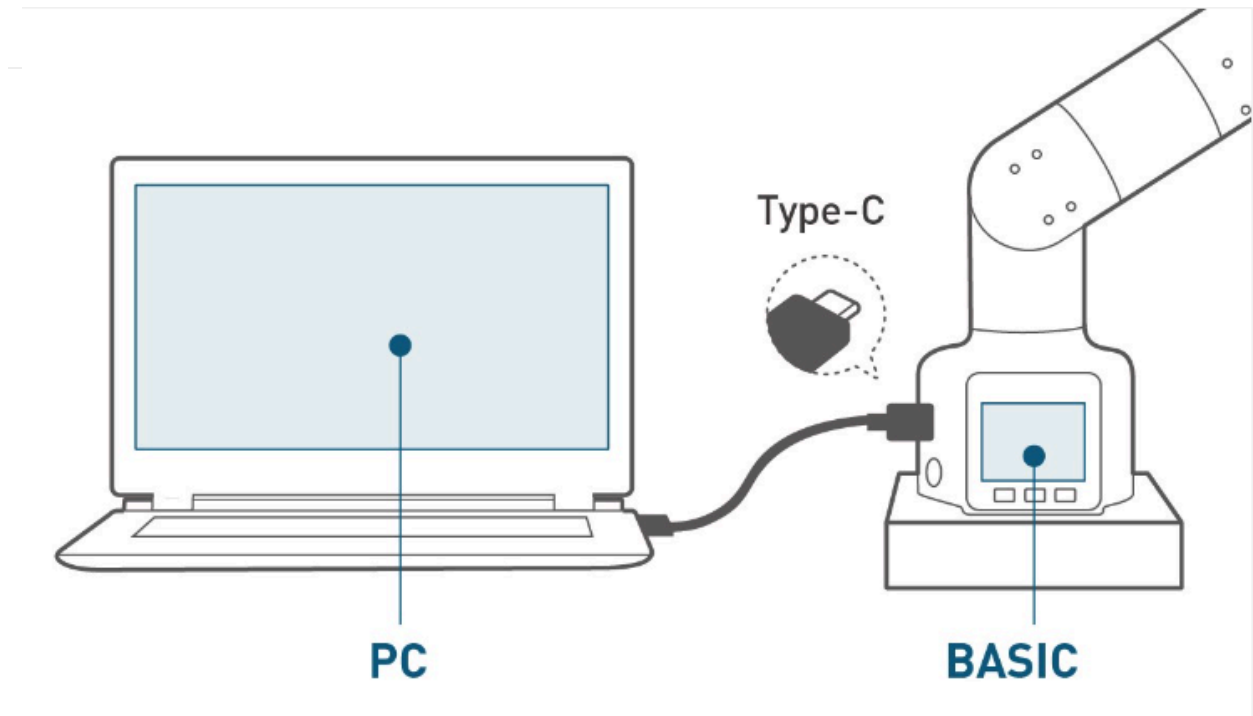


1.3 Connect to another network

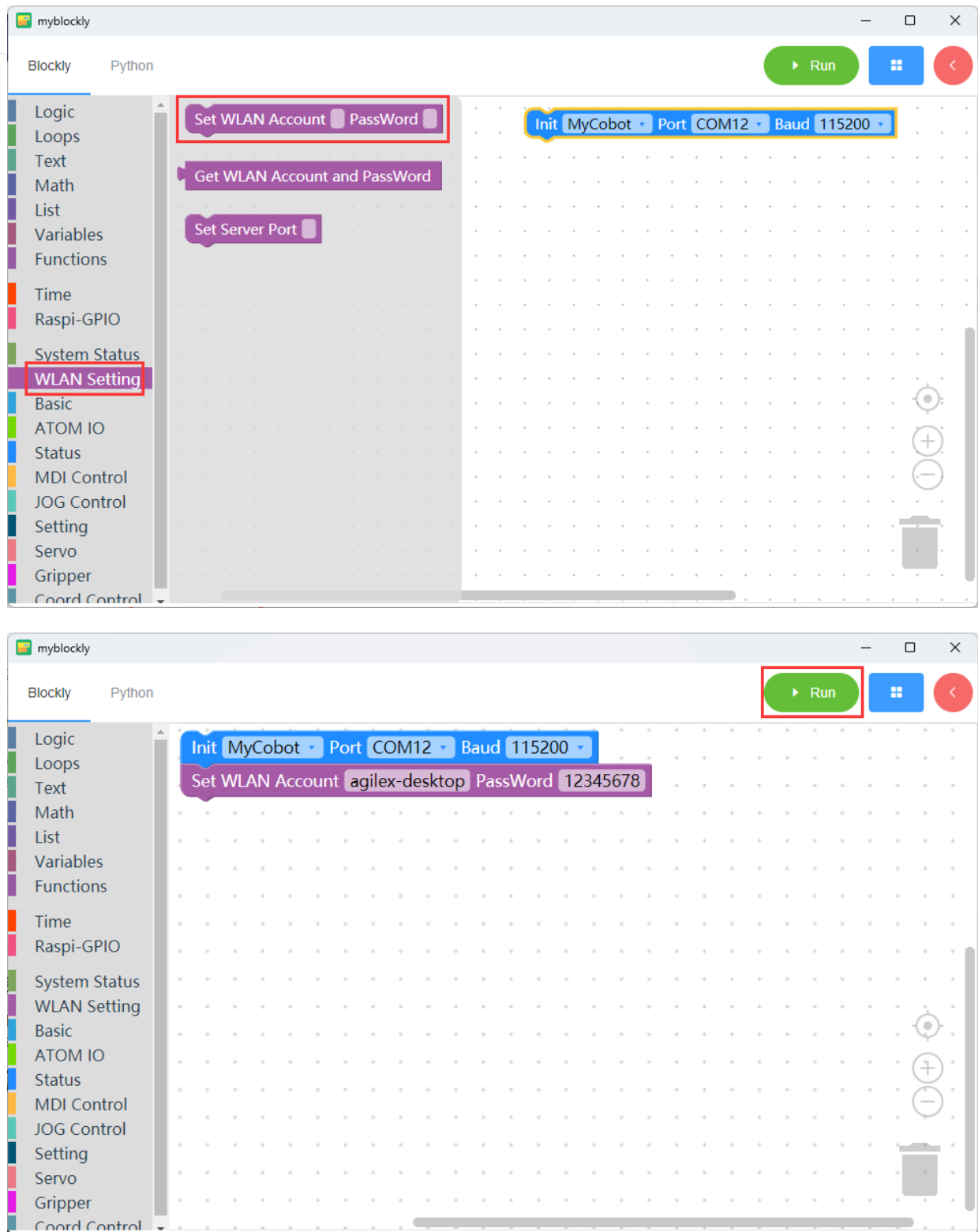
If you need to connect to another network, you can use the [myBlockly](#) software to configure the new network settings.

Note: The myCobot 280 m5 cannot save the connected Wi-Fi account and password when powered off. After a power cycle, it will reconnect to the default Wi-Fi account "MyCobotWiFi2.4G" with the password "mycobot123". To connect to a different network, you will need to reconfigure the Wi-Fi account and password.

Step 1: Connect the PC to the myCobot 280 m5



Step 2: Configure the myCobot 280 m5 Wi-Fi Settings Using myBlockly



Step 3: By following these steps, the myCobot 280 m5 will connect to the "agilex-desktop" network.



1.4 myCobot Raspberry Pi Connections

- When using Raspberry Pi for remote connection, you need to pay attention to the following points
 1. The Raspberry Pi and the control end need to be on the same network
 2. The server file needs to be executed first in the Raspberry Pi (For specific operations, see the gif operation diagram)
 3. After the server file is executed, the prompt "Binding succeeded and waiting connect" indicates that the opening is successful, and the control terminal can refer to **2 Simple Demo**



specific operation is:

clone our project library: `git clone https://github.com/elephantrobotics/pymycobot.git`

find the [Server.py](#) file in the demo folder and execute it with python

1.4 MechArm Raspberry Pi Connections

- When using Raspberry Pi for remote connection, you need to pay attention to the following points
 1. The Raspberry Pi and the control end need to be on the same network
 2. The server file (**change to Server_270.py**) needs to be executed first in the Raspberry Pi (For specific operations, see the gif operation diagram)
 3. After the server file is executed, the prompt "Binding succeeded and waiting connect" indicates that the opening is successful, and the control terminal can refer to **Simple Demo**



specific operation is:

clone our project library: `git clone https://github.com/elephantrobotics/pymycobot.git`

find the [Server_270.py](#) file in the demo folder and execute it with python

Instructions for use:

Please update pymycobot to the latest version before use.

```
pip install pymycobot --upgrade
```

- myPalletizer260

Please change the parameters passed in the last line of the [Server_260.py](#) file, MycobotServer, based on your model.

The default model is the 260PI.

The default parameters are:

```
serial_num: /dev/ttyAMA0
```

```
baud: 1000000
```

- MechArm 270

Please change the parameters passed in the last line of the [Server_270.py](#) file, MechArmSocket, based on your model.

The default model is the 270PI.

The default parameters are:

```
serial_num: /dev/ttyAMA0
```

```
baud: 1000000
```

Simple Demo

When the robotic arm successfully activates the TCP/IP function under the mobile hotspot, it will display the IP address and port. Be sure to remember this IP and port.



Connect your PC to the same mobile hotspot as the robotic arm. By using the Python driver library, you can connect to the robotic arm via its IP address, allowing for remote operation without needing to connect through a USB port.

- myPalletizer260

```
from pymycobot import MyPalletizerSocket

# Use port 9000 by default
# Where "192.168.43.46" is the IP of the robot arm
mc = MyPalletizerSocket("192.168.43.46",9000)

#After the connections is normal, the robot arm can be controlled.
res = mc.get_angles()
print(res)
mc.send_angles([0,0,0,0],20)
...
```

- MechArm 270

```
from pymycobot import MechArmSocket

# Use port 9000 by default
# Where "192.168.43.46" is the IP of the robot arm
mc = MechArmSocket("192.168.43.46",9000)

#After the connections is normal, the robot arm can be controlled.
res = mc.get_angles()
print(res)
mc.send_angles([0,0,0,0,0,0],20)
...
```

myArm

Simple Demo

```
from pymycobot import MyArmSocket

# Use port 9000 by default
# Where "192.168.10.22" is the IP of the robot arm
mc = MyArmSocket("192.168.11.15",9000)

#After the connections is normal, the robot arm can be controlled.
mc.send_angles([0,0,0,0,0,0],20)
res = mc.get_angles()
print(res)

...
```

mybuddy

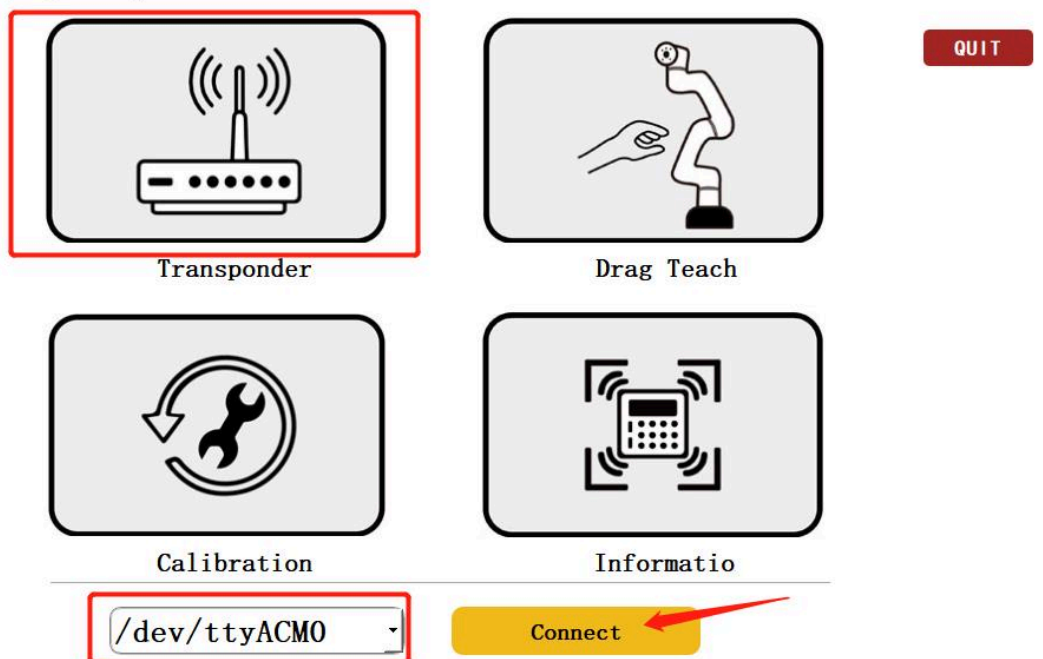
Note: This function must be used under the same network

start the server

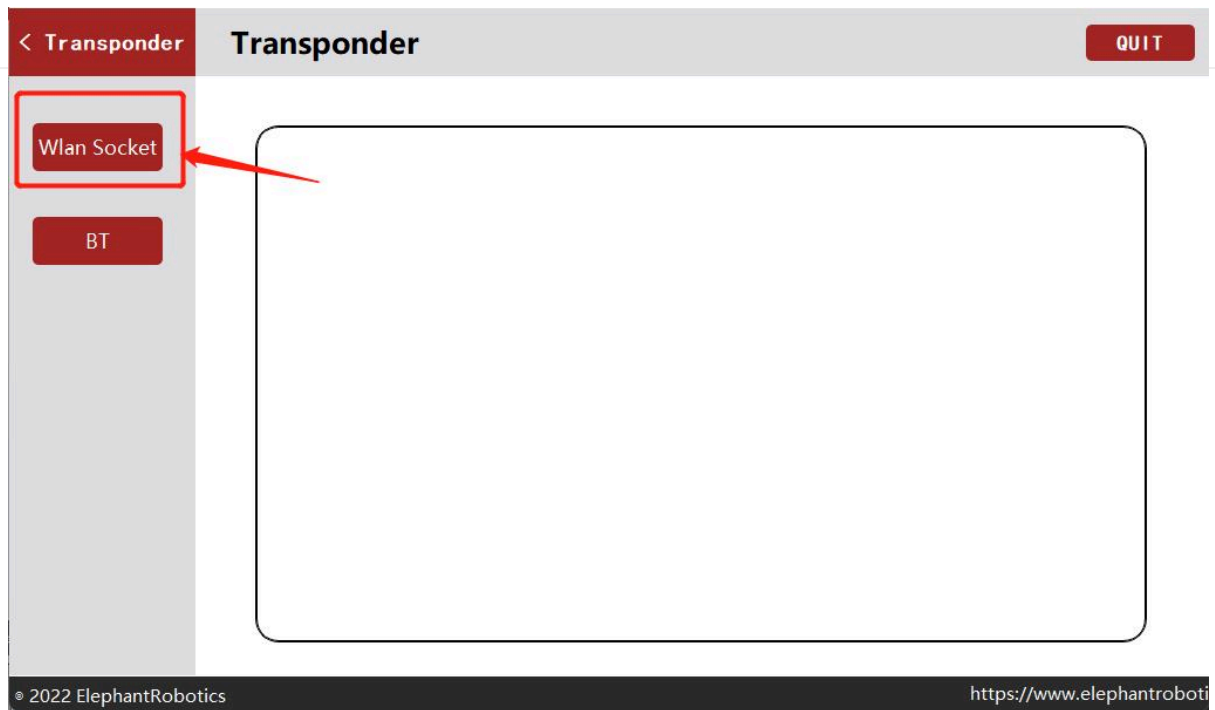
- 1.Double click to open this software on the desktop



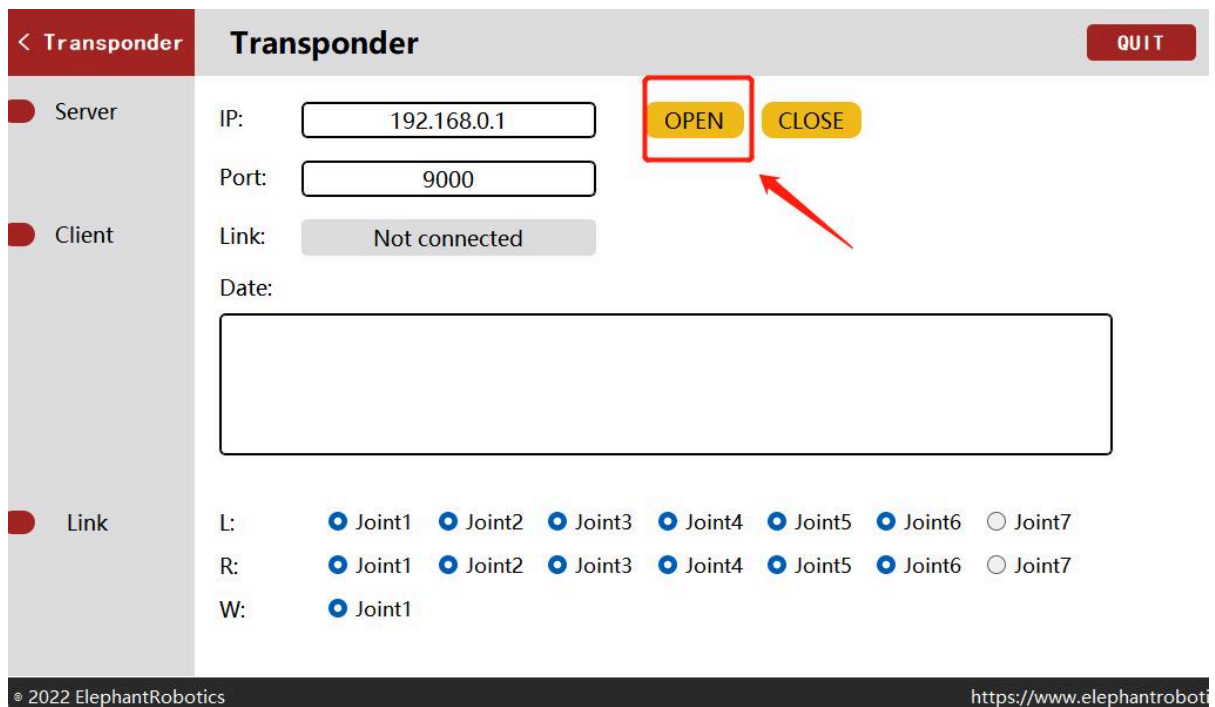
2. Select Transponder, the port is /dev/ttyACM0, click Connect



3. Select "Wlan Socket"



4. Click "OPEN" to open the server



Client connection

```
from pymycobot import MyBuddySocket

mst = MyBuddySocket("192.168.0.1", 9000)
mst.connect("/dev/ttyACM0", "115200")

print(mst.get_angles(1))
```

Drag to teach

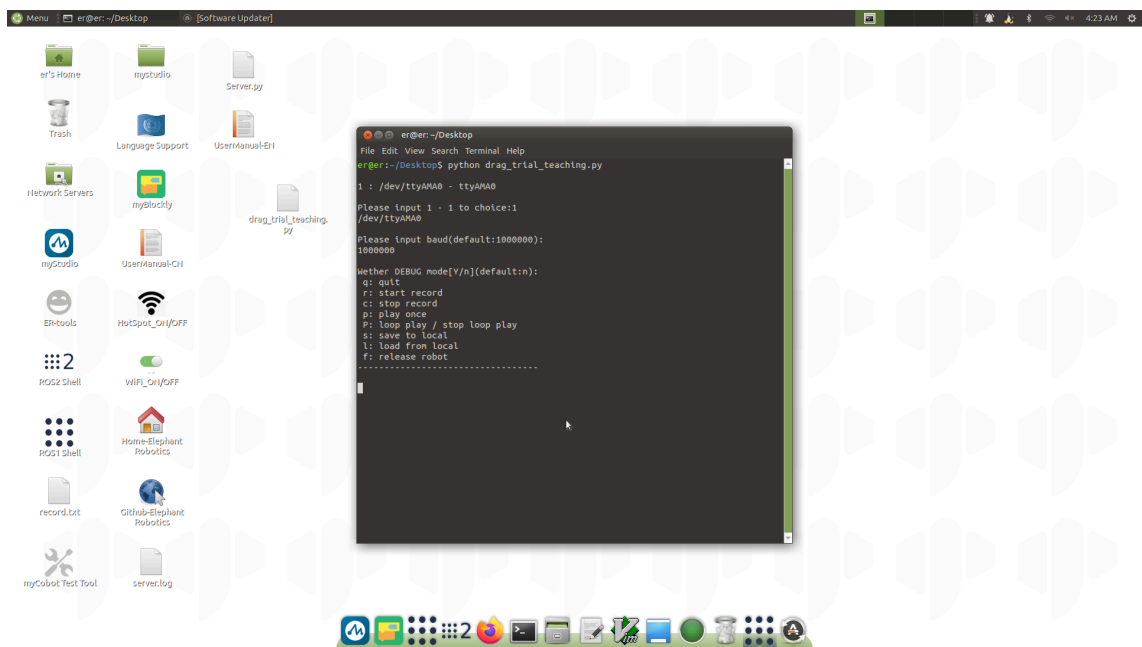
Trajectory recording (including adaptive gripper) and playback can be realized.

Python version drag and teach only supports Pi and Jetson Nano versions in 280, 270, and 320 models.

Instructions:

- 1. Download [Drag Teaching Python Code File](#)
- 1. Copy the downloaded file to the robotic arm system.
- 1. Open the terminal and run the file:

```
python drag_trial_teaching.py
```



After the file is run:

- 3.1 Select robot port
- 3.2 Enter the baud rate, the default is 1000000 | machine type | Serial number | baud rate | | :-----: | :-----: | :-----: | | 270 PI | /dev/ttyAMA0 | 1000000 | 280 PI | /dev/ttyAMA0 | 1000000 | 320 PI | /dev/ttyAMA0 | 115200 | 280 Jetson Nano | /dev/ttyTHS1 | 1000000 |
 - 3.3 Choose whether debugging is required, debugging is not enabled by default
 - 3.4 Finally, enter the function selection list, and the function selection is realized through the keyboard keys:
 - q : quit
 - r : start recording
 - c : stop recording
 - p : play once
 - P : loop / stop playback
 - s : Save this recording information, it will save the recording information to the record.txt file in the same directory as drag_trial_teaching.py
 - l : will load the information in the record.txt file in the same directory as drag_trial_teaching.py
 - f : relax all joints of the robot arm

Videos and Codes for Display

Videos given below are for reference.

Notice: The baud rates are different depending on the type of device. Before using them, refer to the information related thereto. The serial port number can be viewed through [calculator device manager](#) or a serial helper.

1 Controlling RGB Light Panel

MechArm270

```

from pymycobot import MechArm270

from pymycobot import PI_PORT, PI_BAUD      # When using the Raspberry Pi version of mycobot, you can refer to these two v
import time

#The above needs to be written at the beginning of the code, which means importing the project package

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#       or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate:
#       M5 version is: 115200
#
#   Example:
#       mycobot-M5:
#           linux:
#               mc = MechArm270("/dev/ttyUSB0", 115200)
#           or mc = MechArm270("/dev/ttyAMA0", 115200)
#       windows:
#           mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)

i = 7
#loop 7 times
while i > 0:
    mc.set_color(0,0,255) #blue light on
    time.sleep(2)        #wait for 2 seconds
    mc.set_color(255,0,0) #red light on
    time.sleep(2)        #wait for 2 seconds
    mc.set_color(0,255,0) #green light on
    time.sleep(2)        #wait for 2 seconds
    i -= 1

```



myArm

```
from pymycobot.myarm import MyArm
import time

#The above needs to be written at the beginning of the code, which means importing the project package

# Initialize a MyArm object
mc = MyArm("/dev/ttyAMA0", 115200)

i = 7
#loop 7 times
while i > 0:
    mc.set_color(0,0,255) #blue light on
    time.sleep(2)        #wait for 2 seconds
    mc.set_color(255,0,0) #red light on
    time.sleep(2)        #wait for 2 seconds
    mc.set_color(0,255,0) #green light on
    time.sleep(2)        #wait for 2 seconds
    i -= 1
```


2 Controlling Arms to Move Them to Starting Point

MechArm270

```

from pymycobot.mycobot import MechArm270
from pymycobot import PI_PORT, PI_BAUD      # When using the Raspberry Pi version of mycobot, you can refer to these two v

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#       or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate:
#       M5 version is: 115200
#
#   Example:
#       MechArm270-M5:
#       linux:
#           mc = MechArm270("/dev/ttyUSB0", 115200)
#       or mc = MechArm270("/dev/ttyAMA0", 115200)
#       windows:
#           mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code here for Raspberry Pi version
mc = MechArm270(PI_PORT, PI_BAUD)

# Check whether the program can be burned into the robot arm
if mc.is_controller_connected() != 1:
    print("Please connect the robot arm correctly for program writing")
    exit(0)

# Fine-tune the robotic arm to ensure that all the bayonets are aligned in the adjusted position
# Subject to the alignment of the mechanical arm bayonet, this is only a case
mc.send_angles([0, 0, 0, 0, 0, 0], 30)

```



3 Single-Joint Motion

MechArm270

```

from pymycobot import MechArm270
from pymycobot.genre import Angle
import time

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#       or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate:
#       M5 version is: 115200
#
# Example:
#   mycobot-M5:
#       linux:
#           mc = MechArm270("/dev/ttyUSB0", 115200)
#       or mc = MechArm270("/dev/ttyAMA0", 115200)
#       windows:
#           mc = MechArm270("COM3", 115200)
#   mycobot-raspi:
#       mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MMechArm270 object
# Create object code for Raspberry Pi
# mc = MechArm270(PI_PORT, PI_BAUD)
# Create object code for M5
mc=MechArm270('COM3',115200)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 40)
time.sleep(3)

# Control joint 3 to move 70°
mc.send_angle(Angle.J3.value,60,40)
time.sleep(3)

# Control joint 4 movement -70°
mc.send_angle(Angle.J4.value,-70,40)
time.sleep(3)

# Control joint 1 to move 90°
mc.send_angle(Angle.J1.value,90,40)
time.sleep(3)

# Control joint 5 movement -90°

```

1 myCobot Series 6-Axis Collaborative Robotic Arm

```
mc.send_angle(Angle.J5.value,-90,40)
time.sleep(3)

# Control joint 5 to move 90°
mc.send_angle(Angle.J5.value,90,40)
time.sleep(3)
```



4 Multi-Joint Motion

MechArm270

```
import time
from pymycobot import MechArm270

# MechArm270 class initialization requires two parameters:

# The first is the serial port string, such as:
#     linux: "/dev/ttyUSB0"
#     or "/dev/ttyAMA0"
#     windows: "COM3"
# The second is the baud rate::
#     M5 version is: 115200
#
# Example:
#     mycobot-M5:
#         linux:
#             mc = MechArm270("/dev/ttyUSB0", 115200)
#         or mc = MechArm270("/dev/ttyAMA0", 115200)
#     windows:
#         mc = MechArm270("COM3", 115200)
#     mycobot-raspi:
#         mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code for Raspberry Pi
# mc = MechArm270(PI_PORT, PI_BAUD)
# Create object code for M5
mc=MechArm270('COM3',115200)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
time.sleep(2.5)

# Control different angles of rotation of multiple joints
mc.send_angles([90,45,-90,90,-90,90],50)
time.sleep(2.5)

# Return the robotic arm to zero
mc.send_angles([0,0,0,0,0,0],50)
time.sleep(2.5)

# Control different angles of rotation of multiple joints
mc.send_angles([-90,-45,90,-90,90,-90],50)
time.sleep(2.5)
```



5 Swaying Arms Left and Right

MechArm270

```

from pymycobot import MechArm270
from pymycobot.genre import Angle
from pymycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time

# Initialize a MechArm270 object
mc = MechArm270("COM3", 115200)

# Get the coordinates of the current location
angle_datas = mc.get_angles()
print(angle_datas)

#By passing the angle parameter, let each joint of the robotic arm move to the position
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
print(mc.is_paused())

# Set the waiting time to ensure that the robotic arm has reached the specified position
# while not mc.is_paused():
time.sleep(2.5)

# Move joint 1 to the 90 position
mc.send_angle(Angle.J1.value, 90, 50)

# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# set loop times
num = 5

# The following code can make the robotic arm swing left and right
while num > 0:
    # Move joint 2 to the 50 position
    mc.send_angle(Angle.J2.value, 50, 50)

    # Set the waiting time to ensure that the robotic arm has reached the specified position
    time.sleep(1.5)

    # Move joint 2 to the -50 position
    mc.send_angle(Angle.J2.value, -50, 50)

    # Set the waiting time to ensure that the robotic arm has reached the specified position
    time.sleep(1.5)

    num -= 1

# Make the robotic arm retract. You can manually swing the robotic arm, and then use the get_angles() function to get the

```

1 myCobot Series 6-Axis Collaborative Robotic Arm

```
mc.send_angles([88.68, -70, 60, -128.05, -9.93, -15.29], 50)

# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2.5)

# Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```



6 Let Robot Dance

MechArm270

```

from pymycobot import MechArm270

from pymycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time

if __name__ == "__main__":
    # MechArm270 class initialization requires two parameters:
    #   The first is the serial port string, such as:
    #       linux: "/dev/ttyUSB0"
    #           or "/dev/ttyAMA0"
    #       windows: "COM3"
    #   The second is the baud rate::
    #       M5 version is: 115200
    #
    #   such as:
    #       mycobot-M5:
    #           linux:
    #               mc = MechArm270("/dev/ttyUSB0", 115200)
    #           or mc = MechArm270("/dev/ttyAMA0", 115200)
    #           windows:
    #               mc = MechArm270("COM3", 115200)
    #       mycobot-raspi:
    #           mc = MechArm270(PI_PORT, PI_BAUD)
    #
    # Initialize a MechArm270 object
    # Create object code for Raspberry Pi version below
    mc = MechArm270(PI_PORT, PI_BAUD)

    # set start time
    start = time.time()

    # Let the robotic arm reach the specified position
    mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)

    # Determine if it reaches the specified position
    while not mc.is_in_position([-1.49, 115, -153.45, 30, -33.42, 137.9], 0):
        # Return the robotic arm to motion
        mc.resume()

        # Let the robotic arm move for 0.5s
        time.sleep(0.5)

        # Pause arm movement
        mc.pause()

        # Determine if the move timed out
        if time.time() - start > 3:
            break

    # set start time
    start = time.time()

```

1 myCobot Series 6-Axis Collaborative Robotic Arm

```
# Let the exercise last for 30 seconds
while time.time() - start < 30:
    # Let the robotic arm quickly reach this position
    mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
    # Set the color of the light to [0,0,50]
    mc.set_color(0, 0, 50)
    time.sleep(0.7)
    # Let the robotic arm quickly reach this position
    mc.send_angles([-1.49, 55, -153.45, 80, 33.42, 137.9], 80)
    # Set the color of the light to [0,50,0]
    mc.set_color(0, 50, 0)
    time.sleep(0.7)
```



7 Controlling Gripper

MechArm270

```

from pymycobot import MechArm270

from pymycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to the position of 2048
    mc.set_encoder(1, 2048)
    time.sleep(2)

    # Set six joint positions and let the robotic arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(3)

    #Get the position information of joint point 1
    print(mc.get_encoder(1))
    # Set the gripper to rotate to the position of 2048
    mc.set_encoder(7, 2048)
    time.sleep(3)
    # Set the gripper to rotate to the position of 1300
    mc.set_encoder(7, 1300)
    time.sleep(3)

    # Let the gripper reach the state of 2048 at a speed of 70, 2048 will report an error, so change it to 255
    mc.set_gripper_value(255, 70)
    time.sleep(3)
    # Let the gripper reach the state of 1500 at a speed of 70, 1500 will report an error, so change it to 255
    mc.set_gripper_value(255, 70)
    time.sleep(3)

    num=5
    while num>0:
        # Set the state of the gripper to quickly open the gripper at a speed of 70
        mc.set_gripper_state(0, 70)
        time.sleep(3)

```

```

        # Set the state of the gripper to quickly close the gripper at a speed of 70
        mc.set_gripper_state(1, 70)
        time.sleep(3)
        num-=1

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())

if __name__ == "__main__":
    # MechArm270 class initialization requires two parameters:
    #   The first is the serial port string, such as:
    #       linux: "/dev/ttyUSB0"
    #       or "/dev/ttyAMA0"
    #       windows: "COM3"
    #   The second is the baud rate::
    #       M5 version is: 115200
    #
    #   such as:
    #       mycobot-M5:
    #           linux:
    #               mc = MechArm270("/dev/ttyUSB0", 115200)
    #           or mc = MechArm270("/dev/ttyAMA0", 115200)
    #           windows:
    #               mc = MechArm270("COM3", 115200)
    #       mycobot-raspi:
    #           mc = MechArm270(PI_PORT, PI_BAUD)
    #
    # Initialize a MechArm270 object
    # Create object code for Raspberry Pi version below
    mc = MechArm270(PI_PORT, PI_BAUD)
    # make it move to zero position
    mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
    time.sleep(3)
    gripper_test(mc)

```



8 Controlling Sucking Pump

MyCobot 280

280-M5 version (the video below takes the M5 version as an example):

```

from pymycobot import MyCobot280

from pymycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time

# MyCobot280 class initialization requires two parameters:
# The first is the serial port string, such as:
#     linux: "/dev/ttyUSB0"
#     or "/dev/ttyAMA0"
#     windows: "COM3"
# The second is the baud rate::
#     M5 version is: 115200
#
# such as:
#     mycobot-M5:
#         linux:
#             mc = MechArm270("/dev/ttyUSB0", 115200)
#             or mc = MyCobot280("/dev/ttyAMA0", 115200)
#         windows:
#             mc = MyCobot280("COM3", 115200)
#     mycobot-raspi:
#         mc = MyCobot280(PI_PORT, PI_BAUD)
#
# Initialize a MyCobot280 object
# Create object code here for windows version
mc = MyCobot280("COM3", 115200)

# The position of the robot arm movement
angles = [
    [92.9, -10.1, -60, 5.8, -2.02, -37.7],
    [92.9, -53.7, -83.05, 50.09, -0.43, -38.75],
    [92.9, -10.1, -87.27, 5.8, -2.02, -37.7]
]

# Turn on the suction pump
def pump_on():
    # make position 2 work
    mc.set_basic_output(2, 0)
    # make position 5 work
    mc.set_basic_output(5, 0)

# stop the suction pump
def pump_off():
    # Stop position 2 from working
    mc.set_basic_output(2, 1)
    # Stop position 5 from working
    mc.set_basic_output(5, 1)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
time.sleep(3)

```

```
# Turn on the suction pump
pump_on()
mc.send_angles(angles[2], 30)
time.sleep(2)

# absorb small blocks
mc.send_angles(angles[1], 30)
time.sleep(2)
mc.send_angles(angles[0], 30)
time.sleep(2)
mc.send_angles(angles[1], 30)
time.sleep(2)

# Turn off the suction pump
pump_off()
mc.send_angles(angles[0], 40)
time.sleep(1.5)
```

280-Pi version:

```

from pycobot import MyCobot280

from pycobot import PI_PORT, PI_BAUD  ## When using the Raspberry Pi version of mycobot, these two variables can be ref
import RPi.GPIO as GPIO
import time

# MyCobot280 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#       or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#   such as:
#       mycobot-M5:
#           linux:
#               mc = MyCobot280("/dev/ttyUSB0", 115200)
#           or mc = MyCobot280("/dev/ttyAMA0", 115200)
#           windows:
#               mc = MyCobot280("COM3", 115200)
#       mycobot-raspi:
#           mc = MyCobot280(PI_PORT, PI_BAUD)
#
# Initialize a MyCobot280 object
# Create object code here for Raspberry Pi version
mc = MyCobot280(PI_PORT, PI_BAUD)

# The position of the robot arm movement
angles = [
    [92.9, -10.1, -60, 5.8, -2.02, -37.7],
    [92.9, -53.7, -83.05, 50.09, -0.43, -38.75],
    [92.9, -10.1, -87.27, 5.8, -2.02, -37.7]
]

# Initialize the suction pump
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

# Turn on the suction pump
def pump_on():
    # Make the 20 work
    GPIO.output(20, 0)
    # Make the 21 work
    GPIO.output(21, 0)

# Stop the suction pump
def pump_off():
    # Stop the 20 from working
    GPIO.output(20, 1)
    # Stop the 21 from working

```



```
GPIO.output(21, 1)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
time.sleep(3)

# Turn on the suction pump
pump_on()
mc.send_angles(angles[2], 30)
time.sleep(2)

# absorb small blocks
mc.send_angles(angles[1], 30)
time.sleep(2)
mc.send_angles(angles[0], 30)
time.sleep(2)
mc.send_angles(angles[1], 30)
time.sleep(2)

# Turn off the suction pump
pump_off()
mc.send_angles(angles[0], 40)
time.sleep(1.5)
```



ROS/ROS2 Introduction

ROS is the abbreviation of robot operating system. **ROS** is a highly flexible software architecture for writing robot software programs.

Notice:

- Currently, **myCobot 280 series**、**myCobot 320 series**、**myPalletizer 260 series**、**mechArm 270 series** support the use of ROS. for specific situation of the development of various equipment, refer to [equipment development](#).
- Burn the corresponding firmware by using [mystudio](#). Burn the minirobot in M5Stack-basic, choose the transponder function, and burn the latest version of atomMain in atom.

ROS Logo :



ROS is of open source and is a post-operating system or secondary operating system for controlling robots. It provides the functions similar to those provided by an operating system, including hardware abstraction description, low-level driver management, execution of common functions, messages transferred between programs, and program release package management. It also provide some tool programs and libraries used to get, create, write and run multi-machine integration programs.

The primary design goal of **ROS** is to improve the code reuse rate in the R&D field of robots. **ROS** is a framework (i.e. "nodes") for distributed processes, which are encapsulated in program and function packages that can be easily shared and distributed. **ROS** also supports a federated system similar to a code repository, and this system also enables the collaboration and distribution of a project. This design enables the development and realization of a project to to be decided completely independently from the file system to the user interface (no limit by **ROS**). At the same time, all projects can be integrated with the basic tools of **ROS**.

Due to the following disadvantages of **ROS**:

- Limited real-time communication
- System stability does not yet meet industrial-grade requirements
- No security measures
- Only supports Linux(ubuntu)
- The performance of the core mechanism is not optimized to occupy resources

Therefore, **ROS** cannot really enter the industry, and naturally cannot be commercialized. To solve this problem, the community proposed **ROS 2**. It makes **ROS** have the characteristics of productization, including real-time performance, adaptability to all platforms, suitable for hardware with low performance (MCU+RTOS), distributed, data encryption and support for modern programming languages.

ROS2 first removes the master node that exists in **ROS**. After removing the master node, each node can discover each other through the DDS node, each node is equal, and can realize one-to-one, one-to-many, and many-to-many communication. After using DDS for communication, reliability and stability have been enhanced.

Compared with **ROS** that only supports Linux systems, **ROS2** also supports windows, mac, and even RTOS platforms.

Applicable equipment:

- myCobot 280
 - myCobot 280 M5

- myCobot 280 PI
 - myCobot 280 Jetson Nano
 - myCobot 280 for Arduino
-
- myCobot 320
 - myCobot 320 M5
 - myCobot 320 PI
 - myPalletizer 260
 - myPalletizer 260 M5
 - myPalletizer 260 PI
 - myCobot PRO 600
 - mechArm-270
 - mechArm-270 M5
 - mechArm-270 PI

Preconditions for use:

- **M5** series version, the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)
- **Pi \ jetsonnano** series, **ATOM** burns the latest version of **atomMain** (factory default already burnt)

Device Description:

- Among the devices used above, mechArm-270 PI、myCobot 280-Pi, myCobot 280-JetsonNano, and myCobot 320-Pi versions have their own Ubuntu (V-18.04 / V-20.04) system and have a built-in development environment, so they can be used directly without setting up management.
- MyCobot 280-M5, myCobot 320-M5, myCobot 280-Arduino, myPalletizer 260, and mechArm-270-M5 versions need to be built in an environment, but in ROS/ROS2, you only need to [build a ROS environment](#) or [build a ROS2 environment](#).

Movelt Introduction

Movelt is currently the most advanced software for movement operations of robot arms and has been used for more than 100 robots. It integrates the latest achievements in motion planning, control, 3D perception, motion control, control and navigation, provides an easy-to-use platform for developing advanced robot applications, and provides an integrated software platform for design, and integration assessment of new robot products in the fields of industry, commerce, R&D, etc.

Movelt Logo :



Applicable equipment:

- myCobot 280
 - myCobot 280 M5
 - myCobot 280 PI
 - myCobot 280 Jetson Nano
 - myCobot 280 for Arduino

- myCobot 320
 - myCobot 320 M5
 - myCobot 320 PI

- myPalletizer 260
 - myPalletizer 260 M5
 - myPalletizer 260 PI

- myCobot PRO 600

- mechArm-270
 - mechArm-270 PI
 - mechArm-270 M5

Preconditions for use:

- **M5** series version, the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)
- **Pi \ jetsonnano** series, **ATOM** burns the latest version of **atomMain** (factory default already burnt)

ROS Introduction

ROS is an open-source meta-operating system used for robots. It provides an operating system with expected services, including hardware abstraction, low-level device control, implementation of common functions, messages transferred between processes, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run codes across computers.

The "graph" of ROS runtime is a loosely coupled peer-to-peer process network based on a ROS communication infrastructure. ROS implements several different communication methods, including a services mechanism based on synchronous RPC-style communication, a topics mechanism based on asynchronous streaming media data, and a parameter server for data storage.

ROS is not a real-time framework, but it can be embedded in real-time programs. Willow Garage's PR2 robot uses a system called `pr2_etherCAT` to send or receive ROS messages in real time. ROS may also be seamlessly integrated with Orocos real-time toolkits.

1 Design goals and characteristics of ROS

Many people ask "what are differences between ROS and other robot software platforms?" This question is difficult to answer. Because ROS is not a framework that integrates most functions or features. In fact, the main goal of ROS is to provide code reuse support for robot R&D. ROS is a framework (i.e. nodes) for distributed processes, which are encapsulated in program and function packages that can be easily shared and distributed. ROS also supports a federated system similar to a code repository, and this system also enables the collaboration and distribution of a project. This design enables the development and realization of a project to be decided completely independently from the file system to the user interface (no limit by ROS). At the same time, all projects can be integrated with the basic tools of ROS.

To support the primary goals of sharing and collaboration, the ROS framework has several other features:

- **Lean:** ROS is designed to be as lean as possible so that codes written for ROS can be used with other robot software frameworks. The inevitable conclusion from this is that ROS can be easily integrated into other robot software platforms: ROS can already be integrated with OpenRAVE, Orocos and Player.
- **ROS insensitive libraries:** The preferred development model of ROS is written with clean library functions that do not depend on ROS.
- **Language independence:** The ROS framework can simply be implemented in any modern programming language. ROS has implemented Python version, C++ version and Lisp version. It also has experimental libraries for Java and Lua versions.
- **Loose coupling:** The function modules in ROS are encapsulated in independent function packages or meta-function packages, which are easy to share. The modules in the function package are run in units of nodes. With ROS standard IO used as the interface, developers do not need to pay attention to the internal implementation of the module, as long as they understand the interface rules, they can achieve reuse and point-to-point loose coupling between modules.
- **Convenient testing:** ROS has a built-in unit/integration testing framework called `rostopic`, which can easily install or uninstall test modules.
- **Scalable:** ROS is applicable to large runtime systems and large development processes.
- **Free and open-source:** It has many developers and many function packages.

2 Why ROS is used?

Through ROS, we can realize the simulated control of robot arms in a virtual environment.

we will visualize robot arms through **rviz**, operates our robot arms in a variety of ways, and plan and executes the action path of robot arms through **Movelit** to achieve the effect of freely controlling the robot arms.

We will learn how to control our products through the platform in ros in the following chapters.

M5 Version:

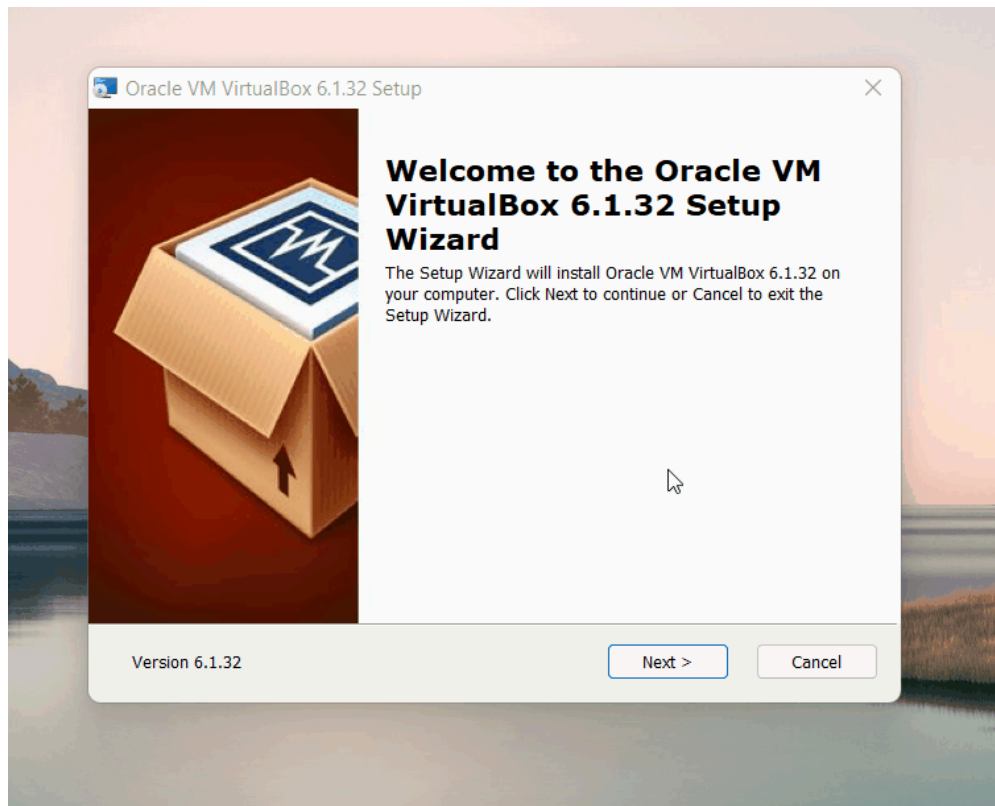
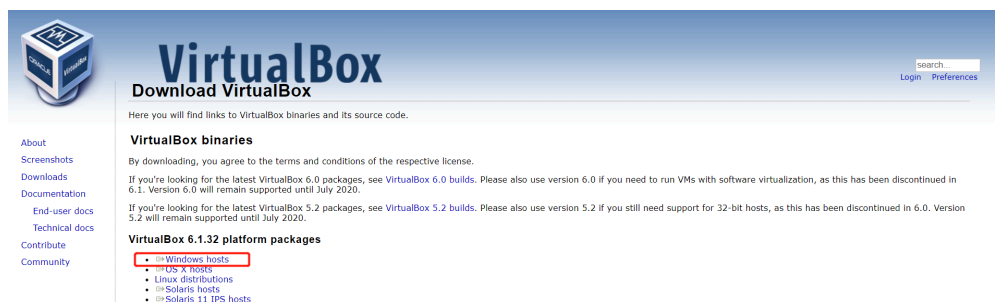
Install different versions of ubuntu system in Linux

1 Virtual machine installation

Go to [Official website](#) to download the virtual machine Virtual Box or go to [Official website](#) to download the virtual machine VMware.

Of course, if you already have your virtual machine, you may skip this step.

We chose to download the Virtual box because it is free.

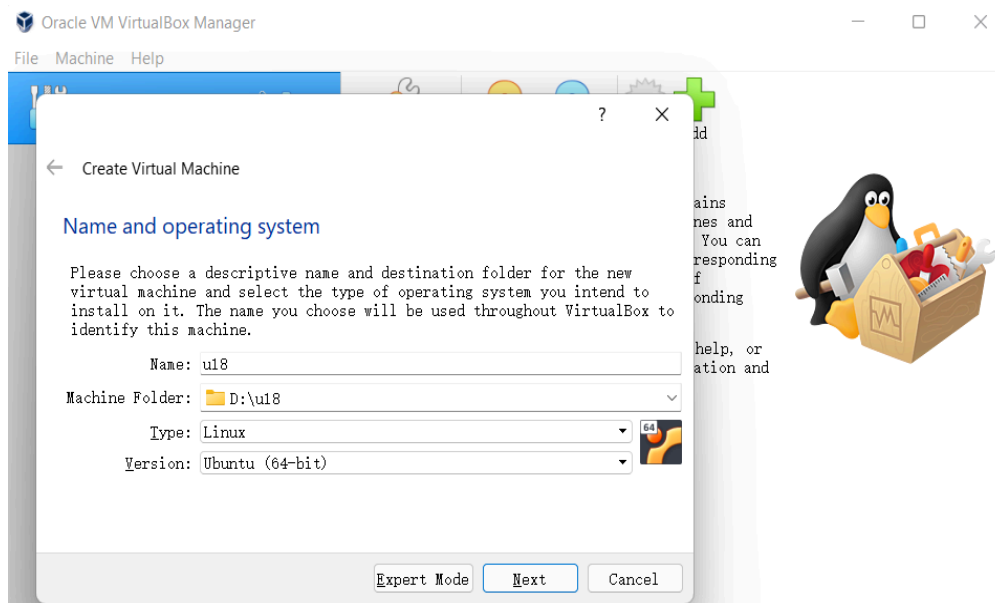


2 Creating a virtual machine

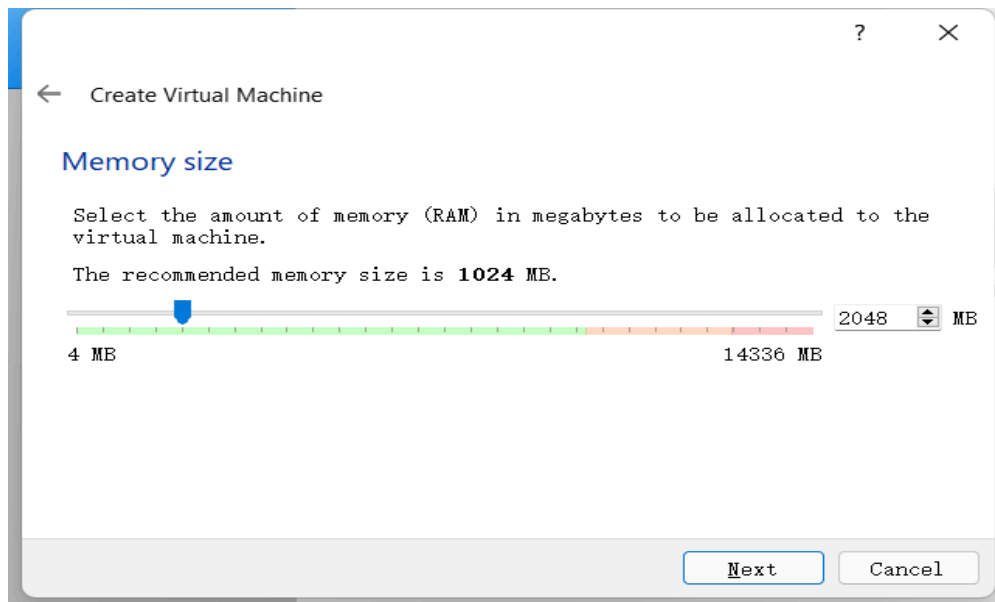
2.1 Creating a virtual machine

Select Create in control

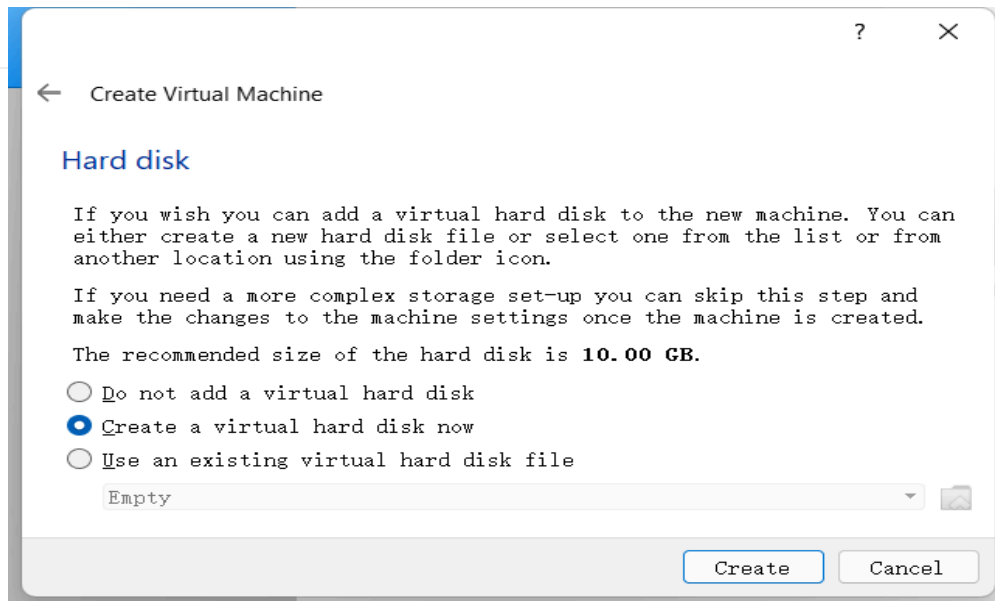
- Input the name of the virtual machine and the location where it is stored, select the type of virtual machine as Linux, select the ubuntu64-bit version, and go to the next step.



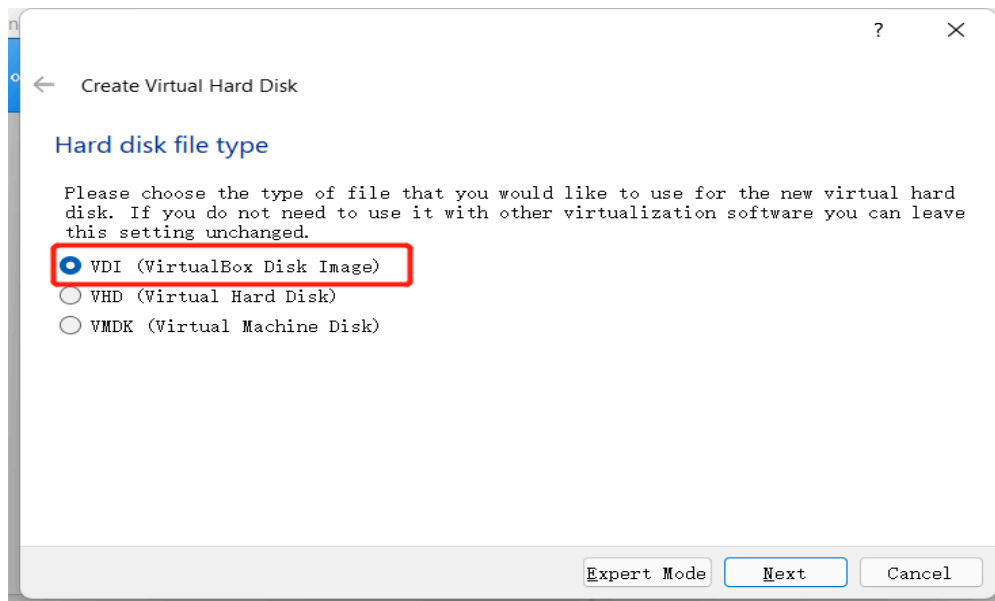
- Configure a memory size according to your own needs, and then go to the next step.



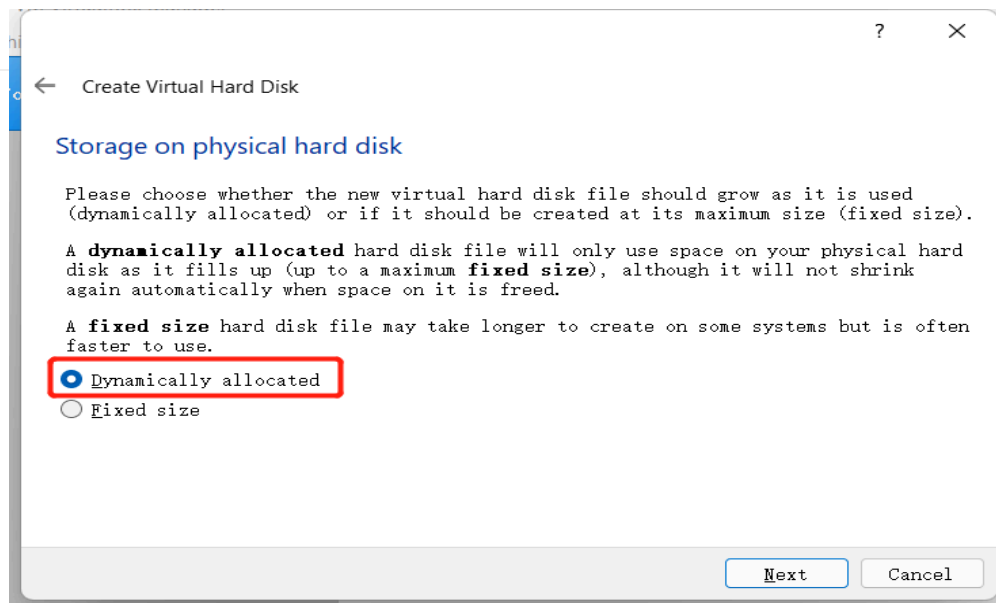
- Choose **Create a virtual hard disk now** to create it.



- Select the virtual hard disk type as **VDI** type, and go to the next step.



- Allocate the size of the virtual hard disk. Since an ubuntu system needs to be installed, and the operation will be performed in the system, it is recommended that the size should not be less than 20G.



Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.



3 Importing the ubuntu system

3.1 Downloading the ubuntu system

Select the ubuntu version according to your own needs and install it:

Note: ROS2 needs to download **version 20.04**.

- [Version 16.04](#)
- [Version 18.04](#)
- [Version 20.04](#)

The installation method and process of the three versions are the same. Here, we take the version 18.04 version for an example.

1 myCobot Series 6-Axis Collaborative Robotic Arm

If you need help burning these images to disk, see the [Image Burning Guide](#).

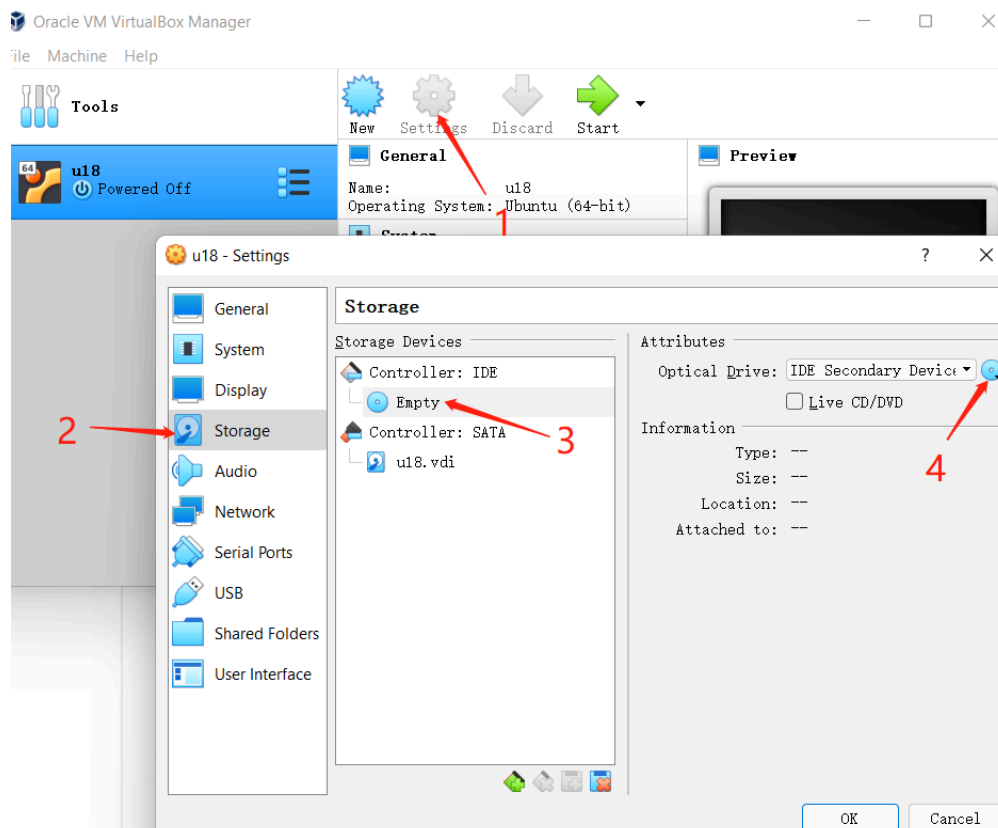
Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS-metalink	2020-02-12 13:42	296	
MD5SUMS-metalink.gpg	2020-02-12 13:42	916	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers (standard download)
ubuntu-18.04.6-desktop-amd64.iso.torrent	2021-09-16 21:46	188K	Desktop image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-18.04.6-desktop-amd64.iso.zsync	2021-09-16 21:46	4.7M	Desktop image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-18.04.6-desktop-amd64.list	2021-09-15 20:42	7.8K	Desktop image for 64-bit PC (AMD64) computers (file listing)
ubuntu-18.04.6-desktop-amd64.manifest	2021-09-15 20:36	59K	Desktop image for 64-bit PC (AMD64) computers (contents of live filesystem)
ubuntu-18.04.6-live-server-amd64.iso	2021-09-15 20:42	969M	Server install image for 64-bit PC (AMD64) computers (standard download)

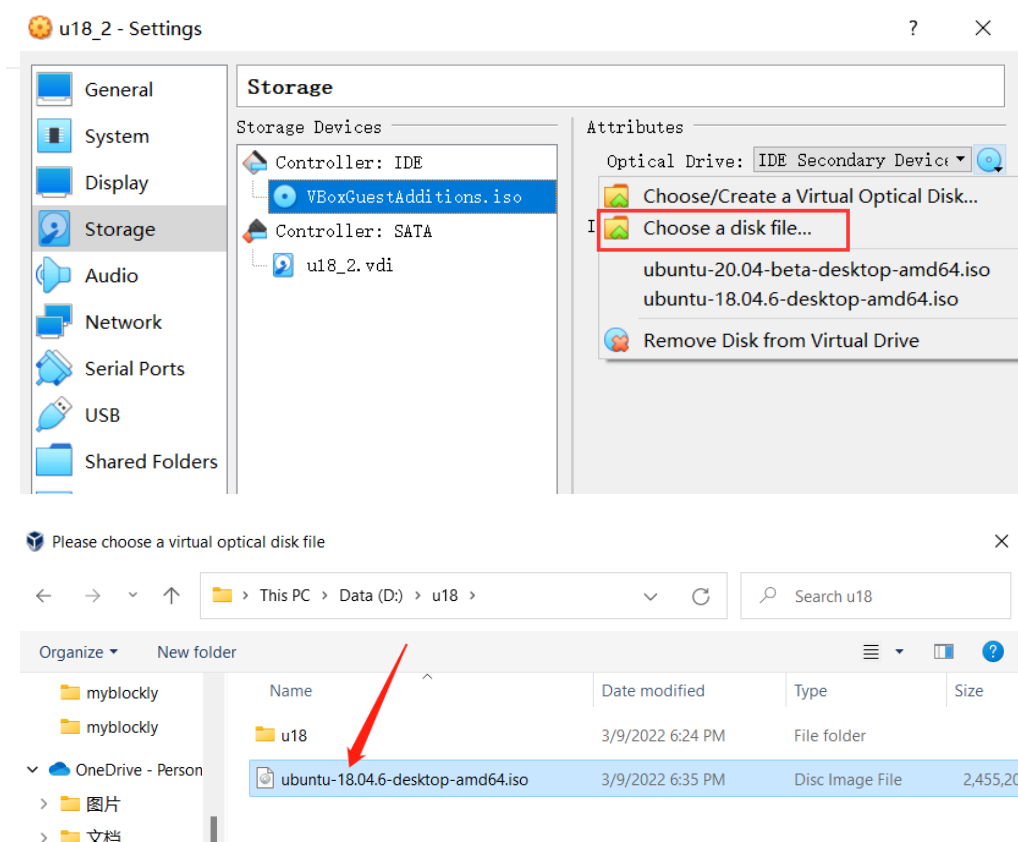
After the downloading is complete, there will be a file shown in the figure:

名称	修改日期	类型	大小
今天 (2)			
ubuntu-20.04.3-desktop-amd64.iso	2022/1/12 15:08	ISO 文件	2,999,93
ubuntu-18.04.6-desktop-amd64.iso	2022/1/12 14:45	ISO 文件	2,455,20

3.2 Importing ubuntu into a visual machine

Find the previously installed virtual machine in the Virtual box, enter **Setup**, and assign a CD to the controller in **Storage**:

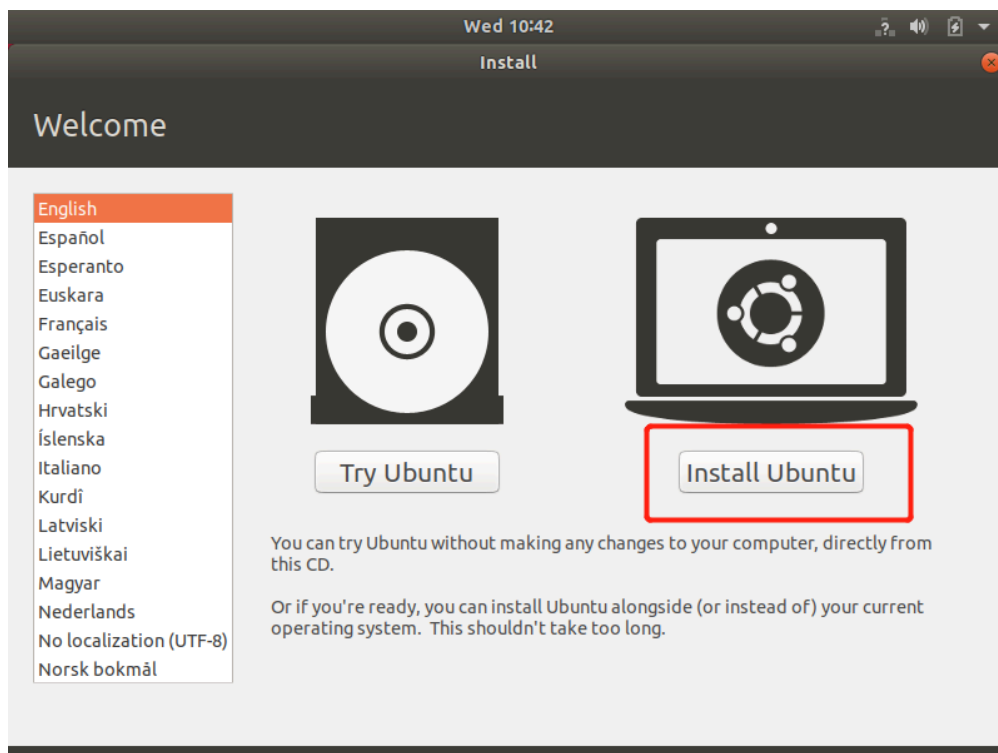




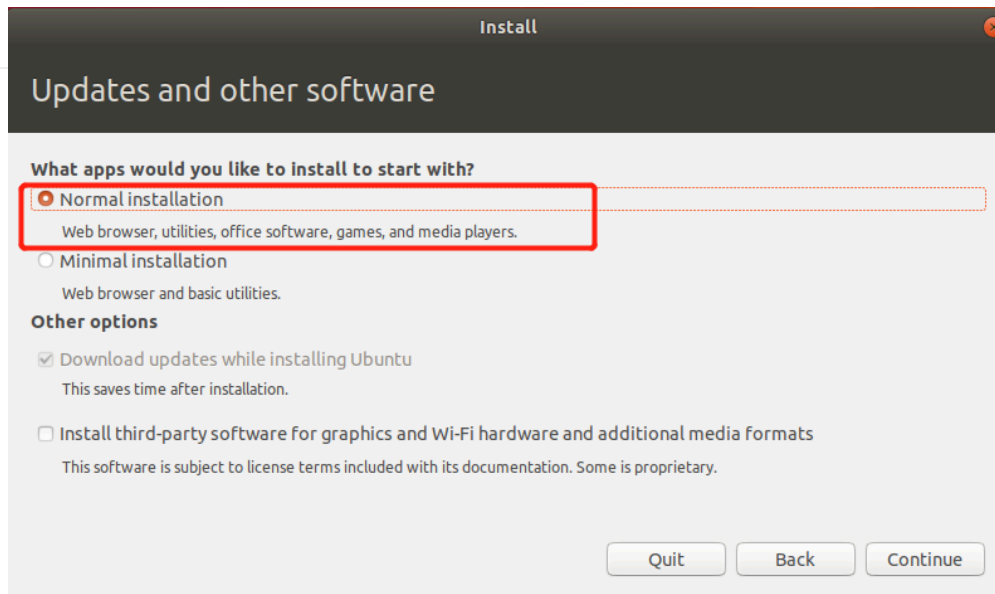
Then open the virtual machine for ubuntu installation, and click start.

3.3 Installing ubuntu

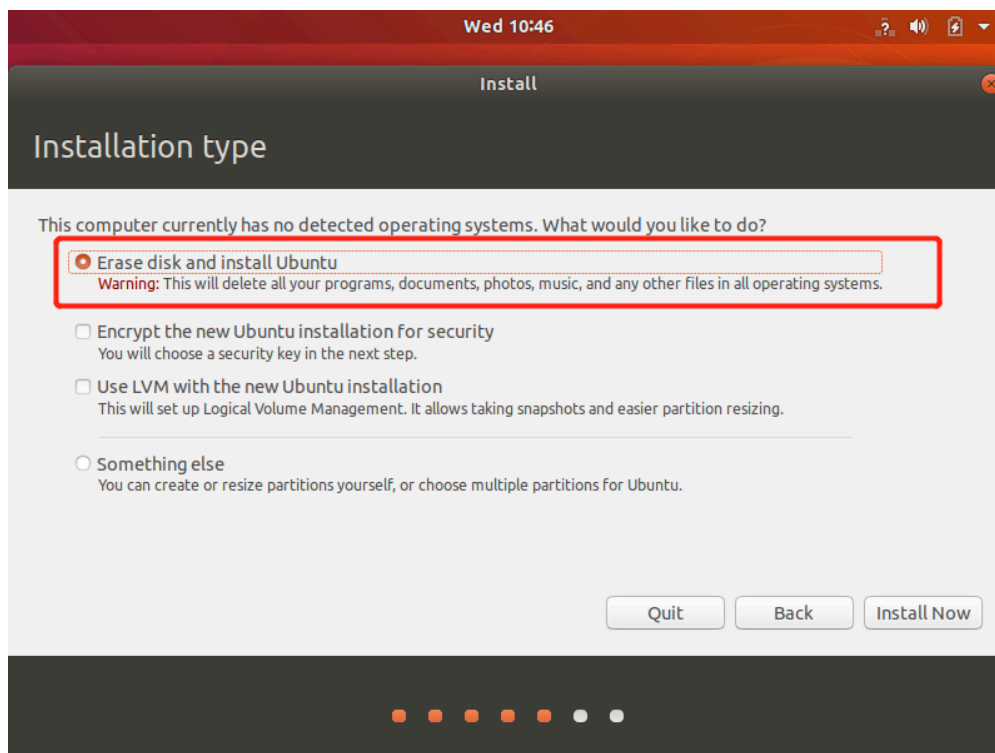
- Wait for the system to start, enter a **welcome** interface, select "Chinese (Simplified)", and click the "Install Ubuntu" button;



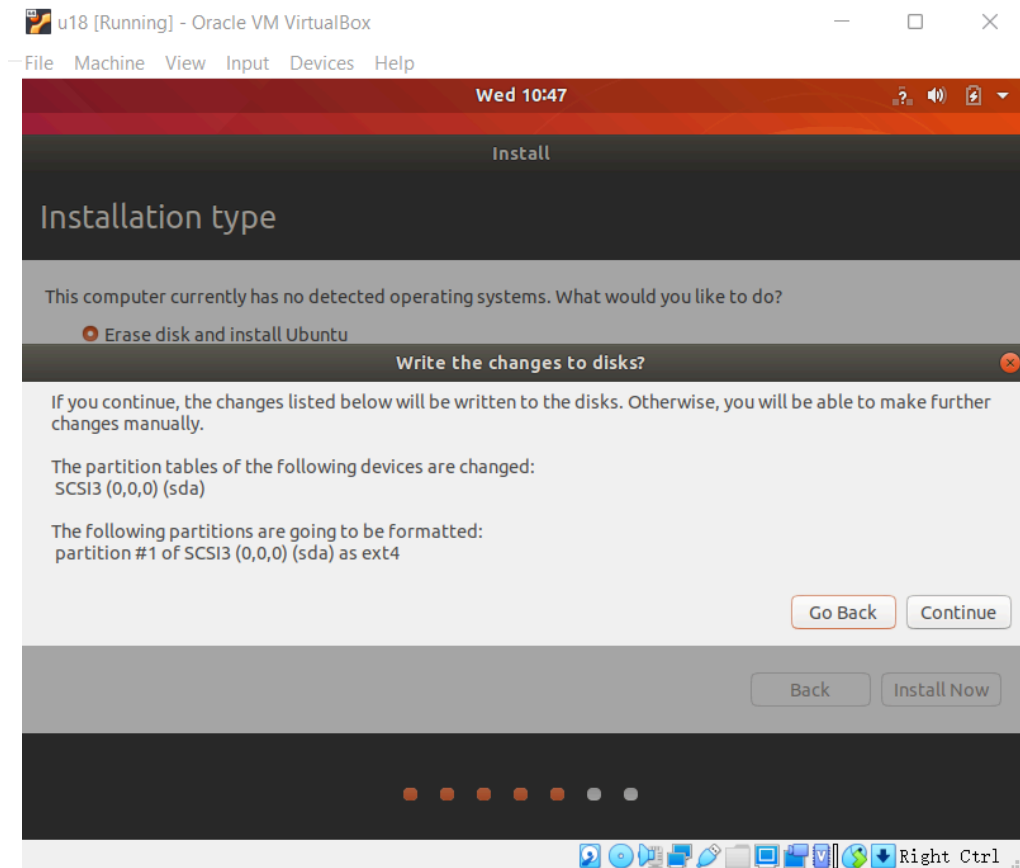
- Click "Continue" button;



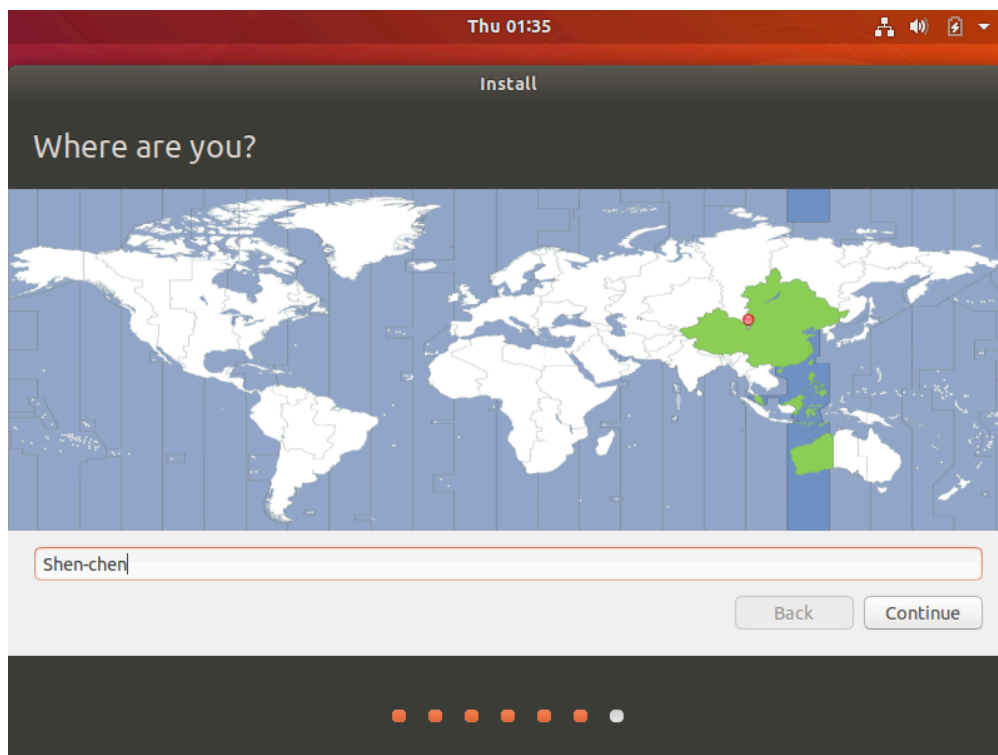
- Select the "Clear entire disk and install Ubuntu" option, and click the "Install Now" button;



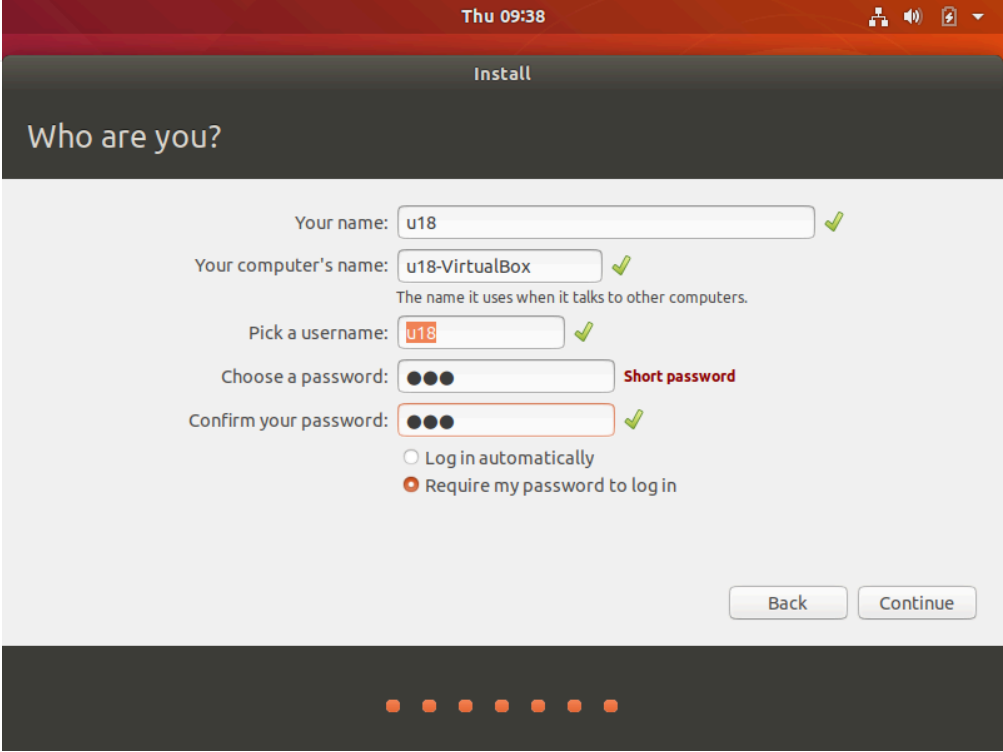
- Click the "Continue" button in the pop-up dialog box;



- Set a geographic position and click the “Continue” button;



- Set a user name and password, and click the “Continue” button;



The image shows the 'Who are you?' screen of the Ubuntu installer. At the top, the window title is 'Install' and the system clock shows 'Thu 09:38'. The main heading is 'Who are you?'. Below this, there are several input fields: 'Your name:' with 'u18' entered and a green checkmark; 'Your computer's name:' with 'u18-VirtualBox' entered and a green checkmark, with a note 'The name it uses when it talks to other computers.'; 'Pick a username:' with 'u18' entered and a green checkmark; 'Choose a password:' with three dots and a red warning 'Short password'; and 'Confirm your password:' with three dots and a green checkmark. At the bottom, there are two radio buttons: 'Log in automatically' (unselected) and 'Require my password to log in' (selected). At the very bottom, there are 'Back' and 'Continue' buttons. A progress bar at the bottom shows six dots, with the first one filled.

Thu 09:38

Install

Who are you?

Your name: u18 ✓

Your computer's name: u18-VirtualBox ✓
The name it uses when it talks to other computers.

Pick a username: u18 ✓

Choose a password: ●●● Short password

Confirm your password: ●●● ✓

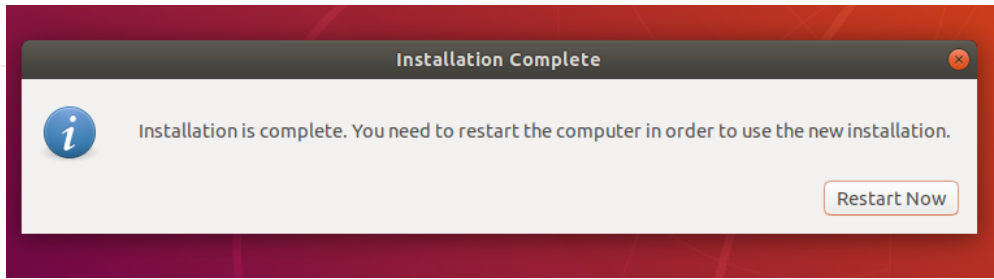
☐ Log in automatically
☒ Require my password to log in

Back Continue

- Enter the system installation interface, and wait patiently;



- After installation is complete, click the "Restart now" button in the pop-up dialog box to complete the installation.



1 ROS Environment building

1.1 Installing ROS

Building a basic development environment requires the installation of the robot operating system (short for ROS), MoveIt and a git version manager. The installation methods and procedures are described respectively below.

For **myCobot 280-M5** and **myCobot 320-M5 devices**, refer to the installation methods and procedures described below. For **myCobot 280-PI** and **myCobot 320-PI** devices, you only need to install the mycobot_ros installation package.

1.1.1 Selecting a version

There is a one-to-one relationship between ROS and ubuntu. Different versions of ubuntu correspond to different versions of ROS. The reference website is as follows: <http://wiki.ros.org/Distributions>

- Here are the ROS versions supported by Ubuntu:
 - Ubuntu 16.04 / ROS Kinetic
 - Ubuntu 18.04 / ROS Melodic
 - Ubuntu 20.04 / ROS Noetic

Install the ROS version corresponding to the Ubuntu version you have installed.

If the versions are different, the downloading will fail. The system we choose here is Ubuntu 18.04, so the corresponding ROS version is ROS Melodic.

NOTE: At present, we do not provide any reference for installing ROS in windows. If necessary, refer to <https://www.ros.org/install/>

1.1.2 Installing ROS

1 Adding a software source

There is no ROS software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

- official source:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.lis
```

- If the download speed is slow, it is recommended to select a nearby mirror source to replace the above command. For example, Tsinghua University is:


```
sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ `lsb_release -cs` main" > /etc
```

Here you will be asked to input a user password. Just input the user password set when Ubuntu is installed.

2 Setting a key

Configuring a public key. This step is to let the system confirm that our path is safe, so that there is no problem in downloading the file. Otherwise it will be deleted immediately after downloading:

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

The execution result is displayed as follows:

```
u18@u18-VirtualBox:~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
Executing: /tmp/apt-key-gpghome.ncwvc7DwDj/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: key F42ED6FBAB17C654: public key "Open Robotics <info@osrfoundation.org>" imported
gpg: Total number processed: 1
gpg: imported: 1
u18@u18-VirtualBox:~$
```

3 Installation

After adding a new software source, you need to update the software source list. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

```
sudo apt-get update
```

Execute **Installing ROS**. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command according to your Ubuntu version:

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

It is recommended to install the complete ROS here to prevent the lack of libraries and dependencies.

The installation process takes a long time, so please wait with patience.

- If the following error message appears on the console terminal during the installation process, you need to replace the software source list in /etc/apt/sources.list.

```
Fetch: 325 MB in 2min 34s (3,401 KB/s)
E: Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/universe/w/wxwidgets
3.0/libwxbase3.0-0v5_3.0.4+dfsg-3_amd64.deb Undetermined Error [IP: 91.189.91.3
9 80]
```

- Open a console terminal (shortcut key:Ctrl+Alt+T),enter the following command:

```
sudo gedit /etc/apt/sources.list
```

- 将sources.list中的官方软件源全部替换成下面的阿里云软件源：

Ubuntu 16.04 version：

```
deb http://mirrors.aliyun.com/ubuntu/ xenial main
deb-src http://mirrors.aliyun.com/ubuntu/ xenial main

deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main

deb http://mirrors.aliyun.com/ubuntu/ xenial universe
deb-src http://mirrors.aliyun.com/ubuntu/ xenial universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates universe

deb http://mirrors.aliyun.com/ubuntu/ xenial-security main
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main
deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security universe
```

Ubuntu 18.04 version：

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse
```

Ubuntu 20.04 version：

```

deb http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse

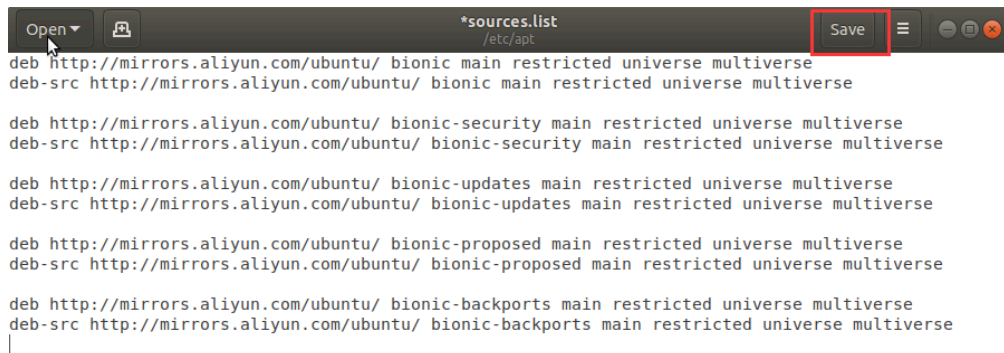
deb http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-proposed main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-backports main restricted universe multiverse

```

-After the configuration is complete, the contents of the sources.list file are as follows, click Save and Exit.



- To update the list of software sources, enter in the console terminal:

```
sudo apt-get update
```

- Enter the instructions to install ROS in the console terminal:

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

The installation process takes a long time, so please wait with patience.

4 Configuring the ROS environment to the system

rosdep allows you to easily install the source codes you want to compile, or the system dependencies required by some ROS core components. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: `Ctrl+Alt+T`).

If your system does not have rosdep installed, use the command `sudo apt install python-rosdep` to install it.

If the system of the Ubuntu installed by you is version 20.04, use the command `sudo apt install python3-rosdep` to it install. After completion, execute the rosdep initialization command.

```
u18@u18-VirtualBox:~$ sudo rosdep init
[sudo] password for u18:
sudo: rosdep: command not found
u18@u18-VirtualBox:~$ sudo apt install python-rosdep
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  docutils-common docutils-doc libpython-stdlib python python-catkin-pkg
  python-catkin-pkg-modules python-chardet python-dateutil python-docutils
  python-minimal python-olefile python-pil python-pkg-resources
  python-pygments python-pyparsing python-roman python-rosdep-modules
  python-rosdistro python-rosdistro-modules python-rospkg
  python-rospkg-modules python-setuptools python-six python-yaml python2.7
  python2.7-minimal sgml-base xml-core
Suggested packages:
  python-doc python-tk fonts-linuxlibertine | ttf-linux-libertine
  texlive-lang-french texlive-latex-base texlive-latex-recommended
  python-pil-doc python-pil-dbg ttf-bitstream-vera python-pyparsing-doc
  python-setuptools-doc python2.7-doc binfmt-support sgml-base-doc debhelper
The following NEW packages will be installed:
  docutils-common docutils-doc libpython-stdlib python python-catkin-pkg
  python-catkin-pkg-modules python-chardet python-dateutil python-docutils
  python-minimal python-olefile python-pil python-pkg-resources
  python-pygments python-pyparsing python-roman python-rosdep
  python-rosdep-modules python-rosdistro python-rosdistro-modules
  python-rospkg python-rospkg-modules python-setuptools python-six
  python-yaml python2.7 python2.7-minimal sgml-base xml-core
0 upgraded, 29 newly installed, 0 to remove and 119 not upgraded.
```

initialize rosdep:

```
sudo rosdep init
```

If the following error message appears:

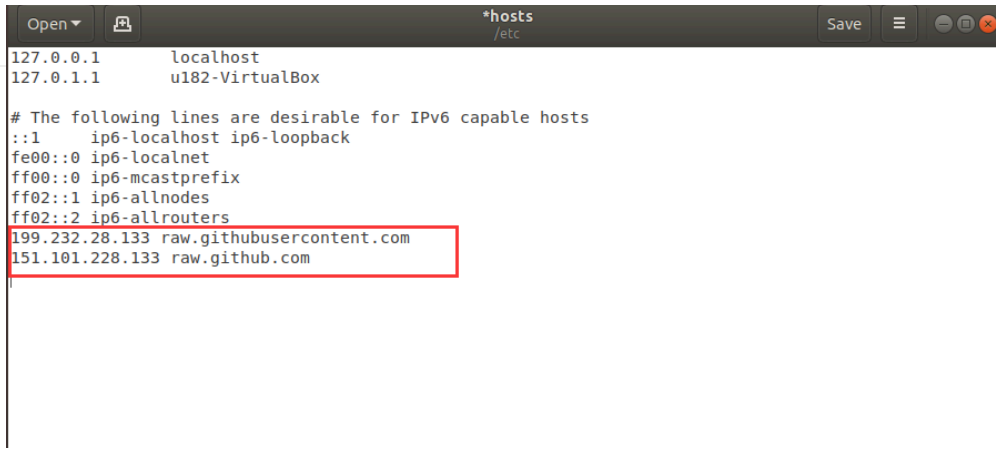
```
u182@u182-VirtualBox:~$ sudo rosdep init
ERROR: cannot download default sources list from:
https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/sources.list.d/20-
default.list
Website may be down.
```

Solution: Modify the hosts file and enter the following command in the console terminal:

```
sudo gedit /etc/hosts
```

At the end of the file content, add the IP addresses of the following two URLs to access:

```
199.232.28.133 raw.githubusercontent.com
151.101.228.133 raw.githubusercontent.com
```



```

Open ▾  *hosts /etc Save ≡ ⌵ ⌵ ⌵ ⌵
127.0.0.1 localhost
127.0.1.1 u182-VirtualBox

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
199.232.28.133 raw.githubusercontent.com
151.101.228.133 raw.githubusercontent.com

```

After the modification is complete, execute in the console terminal:

```
sudo rosdep init
```

```
rosdep update
```

After the initialization is completed, in order to avoid re-validating the ROS function path every time the terminal window is closed, we can **set the path to an environment variable**, so that each time you open a new terminal, the ROS function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

5 Set up the ros environment

Bash

Execute the following commands:

```
# Ubuntu 16.04
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

Zsh

If you replace bash with zsh, then:

```
# Ubuntu16.04
echo "source /opt/ros/kinetic/setup.bash" >> ~/.zshrc
```

```
# Ubuntu18.04  
echo "source /opt/ros/melodic/setup.bash" >> ~/.zshrc
```

```
# Ubuntu20.04  
echo "source /opt/ros/noetic/setup.bash" >> ~/.zshrc
```

```
source ~/.zshrc
```

6 Installing a ROS extra dependency

Input the following command in the terminal to install a ROS extra dependency, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

If your Ubuntu system is version 20.04, please execute the following command to install:

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

```
# Ubuntu 16.04  
sudo apt install ros-kinetic-joint-state-publisher-gui
```

```
# Ubuntu 18.04  
sudo apt install ros-melodic-joint-state-publisher-gui
```

```
# Ubuntu 20.04  
sudo apt install ros-noetic-joint-state-publisher-gui
```

1.1.3 Verifying ROS

The startup of the ROS system requires a ROS Master, that is, a node manager. We can input the `roscore` command in the terminal to start the ROS Master.

To verify whether ROS has been installed successfully, open a console terminal (shortcut key: `Ctrl+Alt+T`) and execute the following command in the terminal:

```
roscore
```

When the following interface is displayed, it means that ROS has been installed successfully.

```

u18@u18-VirtualBox: ~
File Edit View Search Terminal Help
u18@u18-VirtualBox:~$ roscore
... logging to /home/u18/.ros/log/37027a36-751c-11ec-aa4a-0800279b746a/roslaunch
-u18-VirtualBox-1909.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://u18-VirtualBox:37059/
ros_comm version 1.14.12

SUMMARY
=====

PARAMETERS
* /roscore: melodic
* /rosversion: 1.14.12

NODES
auto-starting new master
process[master]: started with pid [1921]
ROS_MASTER_URI=http://u18-VirtualBox:11311/

setting /run_id to 37027a36-751c-11ec-aa4a-0800279b746a
process[roscout-1]: started with pid [1933]
started core service [/roscout]

```

The roscore command starts a node manager, which is used for node management. In a ros system, there is and only one node manager, which is the prerequisite for the operation of the ros node, so before executing the start of the ros node, roscore needs to be executed in the first step.

For more detailed installation instructions, you can refer to the official installation instructions by visiting the website: <http://wiki.ros.org/ROS/Installation>

1.2 MoveIt Installation

MoveIt is the composition of a series of function packages for movement operations in ros, mainly including motion planning, collision detection, kinematics, 3D perception, operation control and other functions.

1.2.1 Updating the software source list

Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window to **update the software source list**:

```
sudo apt-get update
```

1.2.2 Installing MoveIt

Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window to **execute the installation of MoveIt**:

```
# Ubuntu16.04
sudo apt-get install ros-kinetic-moveit
```

```
# Ubuntu 18.04  
sudo apt-get install ros-melodic-moveit
```

```
# Ubuntu20.04  
sudo apt-get install ros-noetic-moveit
```

1.3 git Installation

1.3.1 Adding a software source

Add the software source for installing git to the software source list of ubuntu, open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window:

```
sudo add-apt-repository ppa:git-core/ppa
```

1.3.2 Updating the software source list

Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window to **update the software source list**:

```
sudo apt-get update
```

1.3.3 Installing git

Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window to **execute the installation of git**:

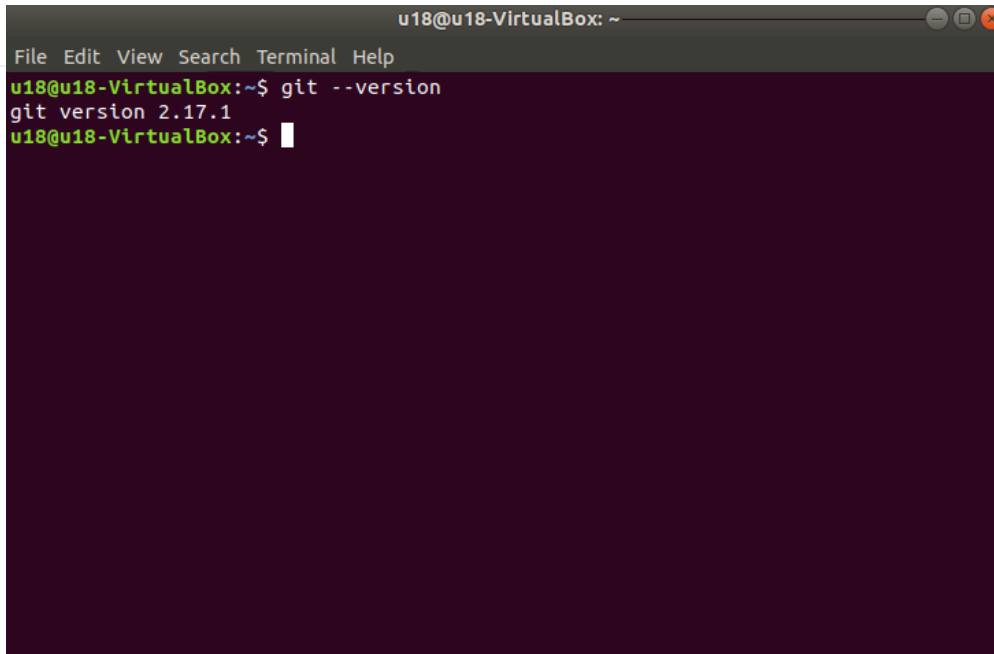
```
sudo apt-get install git
```

1.3.4 Verifying git

Read the git version, open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window:

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, the installation is successful.



```
u18@u18-VirtualBox: ~
File Edit View Search Terminal Help
u18@u18-VirtualBox:~$ git --version
git version 2.17.1
u18@u18-VirtualBox:~$
```

1.3.5 Use

During the subsequent downloading of the ros package, you need to use git. For the use of git, refer to the following link:

- <https://git-scm.com/book/zh/v2>
- <https://www.runoob.com/git/git-tutorial.html>

2 mycobot_ros Installation

`mycobot_ros` is a ROS package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros

2.1 Precondition

Before installing the package, make sure you have a ros workspace.

Here is an **example command for creating a workspace**. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command in the command line:

```
mkdir -p ~/catkin_ws/src # Create a folder
cd ~/catkin_ws/src      # Enter the folder
catkin_init_workspace    # Initialize the current directory into a ROS workspace
cd ..                   # Return to the parent directory
catkin_make              # Build the code in the workspace
```

Add workspace environment

Bash

The official default ROS1 workspace is `catkin_ws` .

```
# Ubuntu 16.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

2.2 Installing the package

NOTE:

- This package depends on ROS and MoveIT. Before using it, make sure that you have installed ROS and MoveIT successfully.
- The interaction of this package with a real robot arm depends on PythonApi - `pymycobot`.
- The Api project location is: <https://github.com/elephantrobotics/pymycobot>
- Quick installation: `pip install pymycobot --upgrade`

When executing the `pip install pymycobot --upgrade` command, if the following error message appears:

```
u182@u182-VirtualBox:~$ pip install pymycobot --upgrade
Command 'pip' not found, but can be installed with:
sudo apt install python-pip
```

Enter the following command to install pip at the prompt

```
sudo apt install python-pip
```

- If your Ubuntu system is version 20.04, please execute the command `sudo apt install python3-pip` to install pip

After the pip installation is complete, the terminal executes again

```
pip install pymycobot --upgrade
```

- The installation method depends on Git, so make sure that Git have been installed in your computer.

The official default ROS1 workspace is `catkin_ws`.

```
cd ~/catkin_ws/src # Enter the src folder of the workspace
# Clone the code on github
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd .. # return to the workspace
catkin_make # Build the code in the workspace
source devel/setup.bash # add environment variables
```

PI Version:

The Raspberry Pi version comes with an Ubuntu (V-20.04) system and a built-in development environment, so you don't need to build and manage it, Just update the mycobot_ros package.

`mycobot_ros` is a ROS1 package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

ROS1 Project address: http://github.com/elephantrobotics/mycobot_ros

Robotic arm API driver library address: <https://github.com/elephantrobotics/pymycobot>

1 Update the mycobot_ros package

In order to ensure that users can use the latest official package in time (new users do not need to update), they can go to the `/home/er/catkin_ws/src` folder through the file manager and open the console terminal (shortcut `Ctrl+Alt+T`), enter the following command to update:

```
# Clone the code on github
cd ~/catkin_ws/src
git clone https://github.com/elephantrobotics/mycobot_ros.git # Please check the attention section below before deciding w
cd .. # Back to work area
catkin_make # Build the code in the workspace
source devel/setup.bash # add environment variable
```

Note: If the `mycobot_ros` folder already exists in the `/home/er/catkin_ws/src` (equivalent to `~/catkin_ws/src`) directory, you need to delete the original `mycobot_ros` before executing the above command. Among them, ubuntu in the directory path is the user name of the virtual machine. If it is inconsistent, please modify it.

So far, the ROS1 environment construction has been completed. Please refer to [13.1.3 Rviz Introduction and Use](#) for the use of ROS1.

1 Service and Topic

We provide some services and topics to interact with mycobot.

1.1 Service

Note: myCobot Pro 600 and myCobot Pro 630 devices do not support Service usage.

Service is a request-response communication mechanism. One node can provide a service, and another node can request the service and wait for a response. Services are usually used to perform time-consuming operations or tasks that require interaction, such as performing calculations and obtaining data.

Related commands and instructions:

command	Detailed description
rosservice list	Display active service information
rosservice info [service name]	Display information about the specified service
rosservice type [service name]	Show service type
rosservice find [service name]	Find a service for a specified service type
rosservice uri [service name]	show ROSRPC URI service
rosservice args [service name]	show service parameters
rosservice call [service name] [parameters]	Request service with input parameters

1.2 Topic

Topic is a publish-subscribe communication mechanism. Nodes can publish messages to a topic, or subscribe to a topic to receive messages. Multiple nodes can publish and subscribe to the same topic at the same time to broadcast and receive information. Topic is mainly used for data transmission with low real-time requirements, such as sensor data, status information, etc.

Related commands and instructions:

Command	Detailed description
rostopic list	Display active topic list
rostopic echo [topic name]	Display the message content of the specified topic in real time
rostopic find [type name]	Display threads with messages of the specified type
rostopic type [topic name]	Displays the message type of the specified topic
rostopic bw [topic name]	Display the message bandwidth of the specified topic (bandwidth)
rostopic hz [topic name]	Display the message data publishing cycle of the specified topic
rostopic info [topic name]	Display information about the specified topic
rostopic pub [topic name] [message type] [parameters]	Post a message with the specified topic name

The difference between service and topic:

	topic	service
Synchronization	Asynchronous	Synchronous
communication model	pub/sub	server/client
underlying protocol	ROSTCP/ROSUDP	RPC
Feedback Mechanism	No	Yes
buffer	Yes	No
Real-time	Weak	Strong
Node Relationship	Many-to-Many	One-to-Many
Applicable Scenarios	Data Transmission	Logical Processing

you can go to [service](#) and [topic](#) learn more about the use of these two features

2 Introduction to msg and srv

- msg: The msg file is a simple text file describing the fields of a ROS message. They are used to generate source code for messages in different languages (c++ or python, etc.).
- srv: srv files are used to describe services. It consists of two parts: the request (request) and the response (response). msg files are stored in the msg directory of the package, and srv files are stored in the srv directory.

2.1 rosmmsg

rosmmsg is a command line tool for displaying information about ROS message types.

rosmmsg demo:

```
rosmmsg show      # Show message description
rosmmsg info      # Display message information
rosmmsg list      # list all messages
rosmmsg md5       # Display md5 encrypted message
rosmmsg package   # Display all messages under a feature pack
rosmmsg packages  # List feature packs that contain messages
```

- rosmmsg list will list all msgs in the current ROS
- rosmmsg packages List all packages containing messages
- rosmmsg package List all msgs under a package

```
//rosmmsg package # Package names
rosmmsg package turtlesim
```

- rosmmsg show Show message description

```
//rosmmsg show # message name
rosmmsg show turtlesim/Pose
# result:
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

- rosmmsg info Works the same as rosmmsg show
- rosmmsg [md5](#) A check algorithm to ensure the consistency of data transmission

2.2 rossrv

rossrv is a command-line tool for displaying information about ROS service types, and uses a syntax that is highly similar to rosmmsg.

```
rossrv show      # Display service message details
rossrv info      # Display information about service messages
rossrv list      # List all service information
rossrv md5       # Display md5 encrypted service messages
rossrv package   # Display all service messages under a package
rossrv packages  # Show all packages that contain service messages
```

- rossrv list Will list all srv messages in the current ROS
- rossrv packages List all packages that contain service messages
- rossrv package List all msgs under a package

```
//rosvrv package # Package names  
rosvrv package turtlesim
```

- rosvrv show Show message description

```
//rosvrv show # message name  
rosvrv show turtlesim/Spawn  
# result:  
float32 x  
float32 y  
float32 theta  
string name  
---  
string name
```

- rosvrv info The effect is the same as rosvrv show
- rosvrv md5 Use md5 checksum (encryption) for service data

3 Introduction to URDF

- Unified Robot Description Format, Unified Robot Description Format, abbreviated as URDF. The urdf package in ROS contains a C++ parser for URDF, and URDF files describe robot models in XML format. *URDF cannot be used alone, it needs to be combined with Rviz or Gazebo. URDF is just a file that needs to be rendered into a graphical robot model in Rviz or Gazebo.

3.1 urdf file description

Code example:

Only part of the code is intercepted here for display:

```

<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="mycobot_ai_with" >

  <xacro:property name="width" value=".2" />

  <link name="env">
    <inertial>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="10"/>
      <inertia
        ixx="1.0" ixy="0.0" ixz="0.0"
        iyy="1.0" iyz="0.0"
        izz="1.0"/>
    </inertial>
    <visual>
      <geometry>
        <!-- 0.0 0 -0.04 1.5708 3.14159-->
        <mesh filename="package://mycobot_description/urdf/mycobot/suit_env.dae"/>
      </geometry>
      <origin xyz = "0 0 0.0" rpy = "1.5708 0 -1.5708"/>
    </visual>
  </link>

  <link name="joint1">
    <inertial>
      <origin xyz="0 0 0.1" rpy="0 0 0"/>
      <mass value="0.2"/>
      <inertia
        ixx="1.0" ixy="0.0" ixz="0.0"
        iyy="1.0" iyz="0.0"
        izz="1.0"/>
    </inertial>
    <visual>
      <geometry>
        <!-- 0.0 0 -0.04 1.5708 3.14159-->
        <mesh filename="package://mycobot_description/urdf/mycobot/joint1.dae"/>
      </geometry>
      <origin xyz = "0.0 0 0.02 " rpy = " 0 0 -1.5708"/>
    </visual>
    <collision>
      <geometry>
        <!-- 0.0 0 -0.04 1.5708 3.14159-->
        <mesh filename="package://mycobot_description/urdf/mycobot/joint1.dae"/>
      </geometry>
      <origin xyz = "0.0 0 0.02 " rpy = " 0 0 -1.5708"/>
    </collision>
  </link>

  <joint name="vision_joint" type="fixed">
    <axis xyz="0 0 0"/>

```



```

    <limit effort = "1000.0" lower = "-3.14" upper = "3.14159" velocity = "0"/>
    <parent link="env"/>
    <child link="joint1"/>
    <origin xyz= "0 0 0" rpy = "0 0 0"/>
  </joint>

  <link name="world"/>
  <joint name="fixed" type="fixed">
    <parent link="world"/>
    <child link="env"/>
  </joint>

</robot>

```

It can be seen that the urdf file is not complicated, it is mainly composed of two parts, `link` and `joint`, which are repeated continuously.

3.2 link section

The link element describes a rigid body with inertial, visual features, and collision properties

3.2.1 Attributes

name: The name used to describe the link itself

3.2.2 element

- `<inertial>` (optional)
 - Inertia properties of connecting rods
 - `<origin>` (optional, defaults to identity if not specified)
 - Defines the reference coordinate of the inertial reference system relative to the connecting rod coordinate system. The coordinate must be defined at the center of gravity of the connecting rod, and its coordinate axis may not be parallel to the main axis of inertia.
 - xyz (optional, defaults to zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
 - rpy(optional: defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
 - `<mass>` Mass properties of connecting rods
 - `<inertia>` 3×3 rotational inertia matrix, consisting of six independent quantities: `ixx`, `ixy`, `ixz`, `iyy`, `iyz`, `izz`.
- `<visual>` (optional)
 - Visual properties of the connecting rod. It is used to specify the shape of the link display (rectangle, cylinder, etc.). There can be multiple visual elements in the same link, and the shape of the link is formed by two elements. In general, the model is more complex and can be drawn through solidworks to generate stl calls, and simple shapes such as adding end effectors can be directly written. At the same time, the position of the geometry can be adjusted according to the gap between the theoretical model and the actual model.
 - `<name>` (optional) The name of the connecting rod geometry.
 - `<origin>` (optional, defaults to identity if not specified)
 - The geometry coordinate system relative to the coordinate system of the connecting rod.

- xyz (optional: defaults to zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
 - rpy (optional: defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
-
- `<geometry>` (required)
 - The shape of the visualization, which can be one of the following:
 - `<box>` A rectangle with elements including length, width, and height. The origin is in the center.
 - `<cylinder>` Cylinder, elements include radius and length. center of origin.
 - `<sphere>` Sphere, element containing the radius. The origin is in the center.
 - `<mesh>` The grid, as determined by the file, also provides a scale to define its boundaries. Collada .dae files are recommended, .stl files are also supported, but must be a local file.
 - `<material>` (optional)
 - Visualize the component's material. It can be defined outside the link tag, but it must be inside the robot tag. When defining outside the link tag, the name of the link must be quoted.
 - `<color>` (optional) Color, consisting of red/green/blue/alpha, in the range [0,1].
 - `<texture>` (optional) Material properties, defined by the file.
 - `<collision>` (optional)
 - Collision properties of the link. Collision properties differ from visual properties of connecting rods, and simple collision models are often used to simplify calculations. The same link can have multiple collision attribute labels, and the collision attribute representation of the link is composed of the set of geometric shapes defined by it.
 - `<name>` (optional) Specifies the name of the connecting rod geometry
 - `<origin>` (optional, defaults to identity if not specified)
 - The reference coordinate system of the collision component is relative to the reference coordinate system of the link coordinate system.
 - xyz (optional, default zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
 - rpy (optional, defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
 - `<geometry>` Same as the geometry element description above

Detailed elements and the role of each element can go to [official documentation](#) to view

3.3 joint part

The joint section describes the kinematics and dynamics of the joint and specifies safety limits for the joint.

3.3.1 properties of joint:

name:

Specifies a unique name for the joint

type:

Specifies the type of joint, where type can be one of the following:

- revolute - A hinged joint that rotates along an axis, the range of which is specified by the upper and lower bounds.
- Continuous - A continuous hinged joint that rotates around an axis with no upper and lower bounds.
- Prismatic - A sliding joint that slides along an axis, the range of which is specified by upper and lower limits.
- +Fixed - this is not really a joint because it cannot move. All degrees of freedom are locked. This type of joint

does not require axes, calibration, dynamics, limits or safety_controller.

- Floating - This joint allows motion in all 6 degrees of freedom.
- Plane - This joint allows movement in a plane perpendicular to the axis.

3.3.2 elements of joint

- `<origin>` (optional, defaults to identity if not specified) In the transformation from parent link to child link, the joint is located at the origin of the child link. Modifying this parameter can adjust the position of the connecting rod. It can be used to adjust the error between the actual model and the theoretical model, but it is not recommended to modify it greatly, because this parameter affects the connecting rod stl. The position of , easily affects the collision detection effect.
 - xyz (optional: default to zero vector) Represents the offset in the x , y , z x,y,zx,y,z axis directions, in meters.
 - rpy (optional: default to zero vector) Represents the angle of rotation around a fixed axis: roll is around the x-axis, pitch is around the y-axis, and yaw is around the z-axis, expressed in radians.
- `<parent>` (required)
 - The name of the parent link is a mandatory attribute.
 - link The name of the parent link is the name of the link in the robot structure tree.
- `<child>` (required)
 - The name of the child link is a mandatory attribute.
 - link The name of the child link is the name of the link in the robot structure tree.
- `<axis>` (optional: defaults to (1,0,0))
 - The joint's axis is in the joint's coordinate system. This is the axis of rotation (revolute joint), the axis of movement of the prismatic joint, and the standard plane of the planar joint. This axis is specified in the joint coordinate system. Modifying this parameter can adjust the axis around which the joint rotates. It is often used to adjust the rotation direction. If the model rotation is opposite to the actual one, just multiply by -1. Fixed and floating joints do not need this element.
 - xyz(required) x , y , z x, y, zx, y, z components representing axis vectors, as normalized vectors.
- `<calibration>` (optional)
 - The reference point of the joint, used to correct the absolute position of the joint.
 - rising (optional) When the joint is moving forward, the reference point triggers a rising edge.
 - falling (optional) When the joint is moving forward, the reference point triggers a falling edge.
- `<dynamics>` (optional)
 - This element is used to specify the physical properties of the joint. Its value is used to describe the modeling performance of the joint, especially during simulation.

`<limit>` (Required when the joint is a rotation or translation joint)

- This element is a joint kinematics constraint.
- lower (optional, default to 0) Specify the attribute of the lower bound of the joint's motion range (the unit of the revolute joint is radians, and the unit of the prismatic joint is meters). This attribute is ignored for continuous joints.
- upper (optional, defaults to 0) Specify the attribute of the upper bound of the joint's motion range (the unit of the revolute joint is radians, and the unit of the prismatic joint is the meter). This attribute is ignored for continuous joints.
- effort (required) This property specifies the maximum force at which the joint will run.
- velocity (required) This property specifies the maximum speed of the joint runtime.

`<mimic>` (optional)

- This tag is used to specify a defined joint to mimic an existing joint. The value of this joint can be calculated using the following formula: $\text{value} = \text{multiplier} * \text{other_joint_value} + \text{offset}$
- joint(required) The name of the joint to mimic.
- multiplier(optional) Specify the multiplier factor in the above formula.
- offset(optional) Specify the offset term in the above formula. Default value is 0

`<safety_controller>` (optional)

- This element is a security control limit. The data under this element will be read into move_group, but it is invalid in practice. Move_group will skip this limit and directly read the parameter content under limit. At the same time, setting this element may cause planning failure.
- soft_lower_limit (optional, defaults to 0) This attribute specifies the lower bound of the joint security control boundary, which is the starting limit point of the joint security control. This value needs to be greater than the lower value in the above limit.
- soft_upper_limit (optional, defaults to 0) This attribute specifies the upper bound of the joint security control boundary, which is the starting limit point of the joint security control. This value needs to be less than the upper value in the above limit.
- k_position(optional, defaults to 0) This attribute is used to describe the relationship between position and velocity.
- k_velocity(required) This property is used to describe the relationship between force and velocity.

Detailed elements and the role of each element can go to <http://wiki.ros.org/urdf/XML/joint> to view.

Brief introduction and use of rviz

rviz is a 3D visualization platform in ROS. On one hand, it can realize the graphical display of external information, and on the other hand, it can also release control information to an object through rviz, realizing the monitoring and control of a robot.

1 Installation of rviz and the introduction to its interface

When installing ros, if you perform a complete installation, rviz is already installed, and you may try to run it directly; if it is not fully installed, you may install rviz separately:

```
# Ubuntu16.04  
sudo apt-get install ros-kinetic-rviz
```

```
# Ubuntu18.04  
sudo apt-get install ros-melodic-rviz
```

```
# Ubuntu20.04  
sudo apt-get install ros-noetic-rviz
```

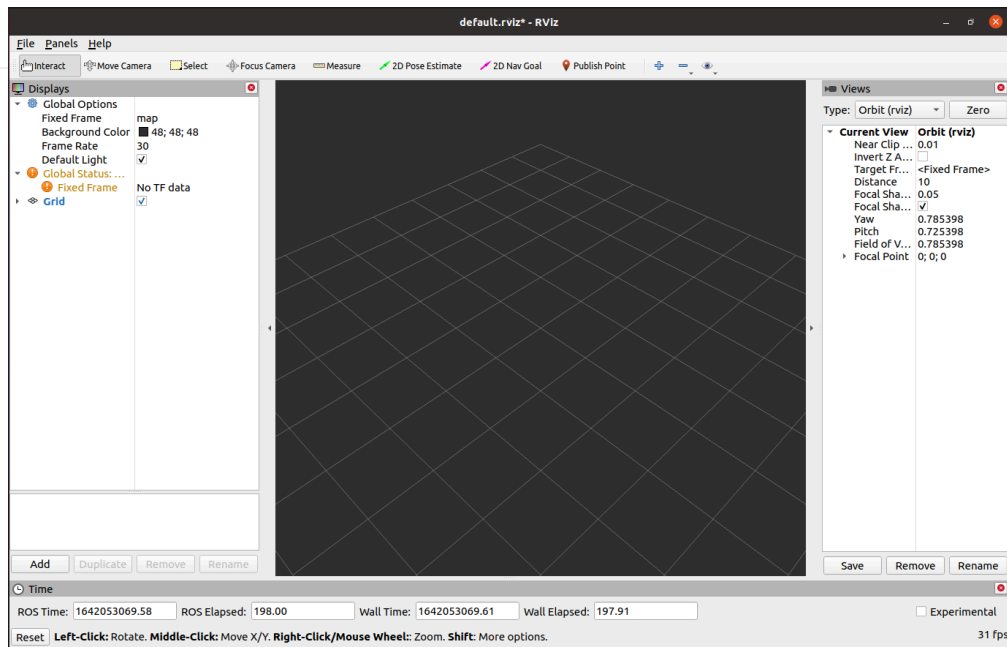
After the installation is complete, open a new terminal (shortcut key: `Ctrl+Alt+T`) and enter the following command:

```
roscore
```

Then open a new terminal (shortcut key: `Ctrl+Alt+T`) and input the following command to open rviz.

```
roslaunch rviz rviz  
# or  
rviz
```

Open rviz, and the following interface will be displayed:



1.1 Introduction of all areas

- There is a list of monitors on the left. The monitor is a device that draws something in a 3D world and may have some options available in the display list.
- On the top is a toolbar, which allows the user to use various function buttons to select tools with multiple functions.
- The middle part is the 3D view: It is a main screen where various data can be viewed in three dimensions. The background color, fixed frame, grid, etc. of the 3D view can be set in detail in the Global Options and Grid items displayed on the left.
- Below is the time display area, including system time and ROS time.
- The right side is the observation angle setting area where different observation angles can be set.

We only give a rough introduction in this part. If you want to know more details, go to [User Guide](#).

2 mycobot_ros installation and update

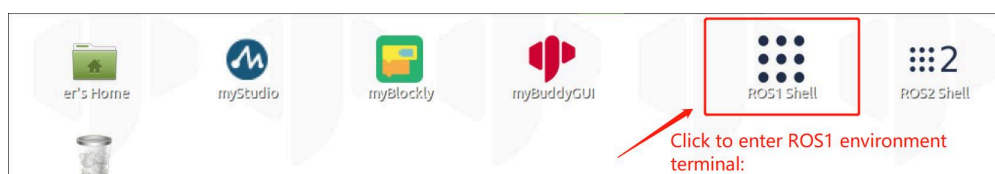
- **M5 Version:** Please refer to the end of the [13.1.1 Environment Building](#) chapter.
- **PI版本(Ubuntu 20.04):**

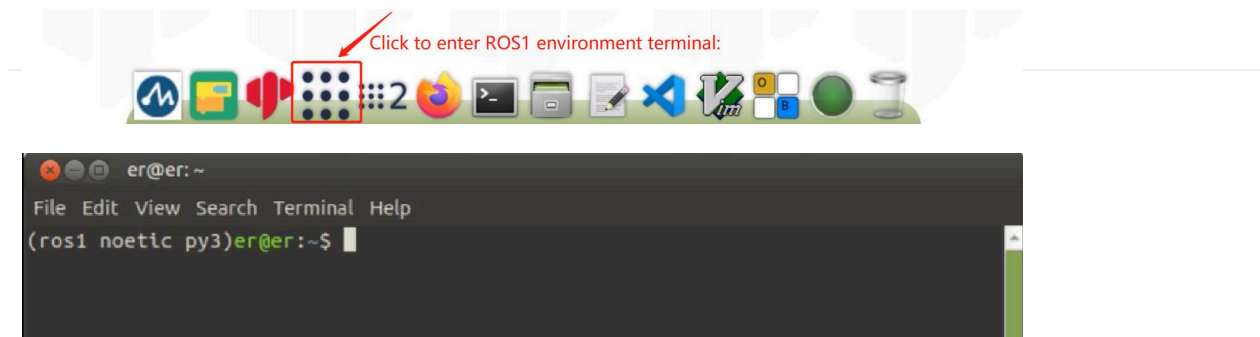
`mycobot_ros` is a ROS package from ElephantRobotics that works with all types of desktop robots.

The address of the project: https://github.com/elephantrobotics/mycobot_ros

The official default ROS1 workspace is `catkin_ws`.

Click the `ROS1 Shell` icon on the desktop or the corresponding icon in the lower bar of the desktop to open the ROS1 environment terminal:





Then enter the following command:

```
cd ~/catkin_ws/src # Enter the src folder of the workspace
# Clone the code on github
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd .. # return to the workspace
catkin_make # Build the code in the workspace
source devel/setup.bash # add environment variables
```

Note: If the `/home/er/catkin_ws/src`(equivalent to `~/catkin_ws/src`) already exists in the `mycobot_ros` folder, you need to delete the `mycobot_ros` folder before running the above command. In the directory path, `er` is the user name of the VM. If they are different, change them.

3 Simple use

Start using launch file

This example is built on what you have already done [Environment building](#) and you have successfully copied the company's code from GitHub to your virtual machine.

Open a new terminal (shortcut key: `Ctrl+Alt+T`)

Input the command to **configure the ROS environment**.

```
cd ~/catkin_ws/
source devel/setup.bash
```

Input again:

- mycobot 280-M5 version:

```
roslaunch mycobot_280 test.launch
```

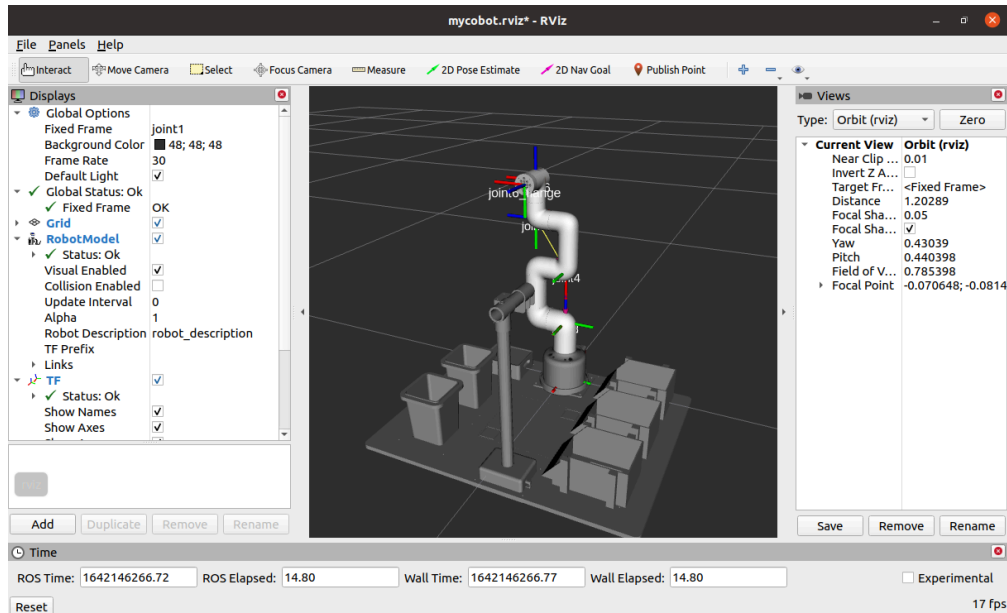
- mycobot 280-Pi version:

```
roslaunch mycobot_280pi test.launch
```

- mycobot 280-JetsonNano version:

```
roslaunch mycobot_280jn test.launch
```

Open rviz, and then you will obtain the following result:



If you want to know more information about rviz, go to [Official documents](#).

4 M5 Version Prerequisites

- Type Ctrl+Shift+T in the command terminal to open another terminal window in the same directory to view the device name:

```
# View the device name of the robotic arm
ls /dev/ttyUSB* # old version myCobot280 M5

# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # new version myCobot280 M5
```

- Grant the serial port permission to the robotic arm:

```
# The default device name is /dev/ttyUSB0, if the device name is not the default value, it needs to be modified.
sudo chmod 777 /dev/ttyUSB0 # old version myCobot280 M5

sudo chmod 777 /dev/ttyACM0 # new version myCobot280 M5
```

Then enter the user password(**Note:** The password is not displayed, just enter it correctly).

Control and following of the robot arm

1 Slider Control

Open a command line and run:

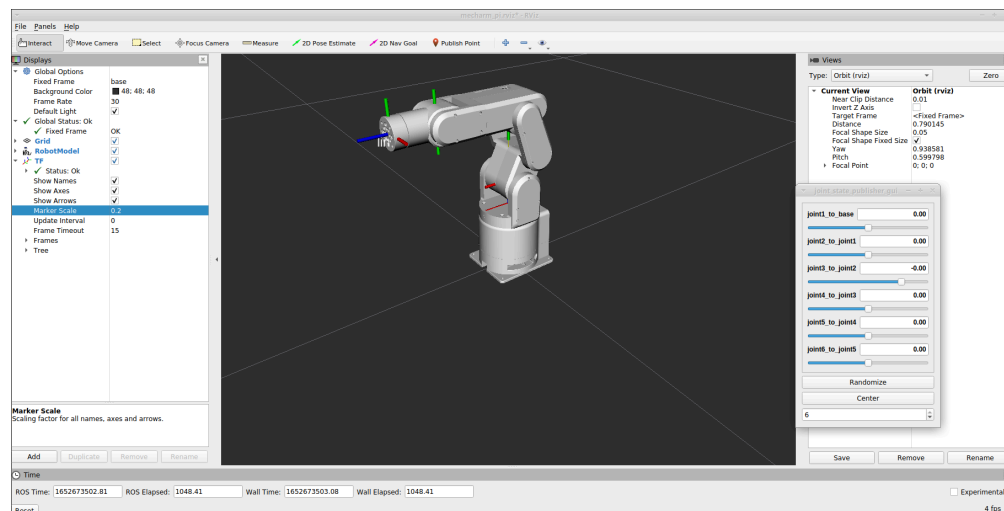
mechArm-M5 version:

```
# The default serial port name of the mechArm-M5 version is "/dev/ttyUSB0", and the baud rate is 115200". The serial port
roslaunch mecharm slider_control.launch port:=/dev/ttyUSB0 baud:=115200
```

mechArm-PI version:

```
# The default serial port name of the mechArm-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".
roslaunch mecharm_pi slider_control.launch port:=/dev/ttyAMA0 baud:=1000000
```

rviz and a **slider** component will be opened, and you will see the following interface:



Then you can **control the model in rviz to make it move by dragging the slider**. If you want the real mechArm to move with the model, you need to open another command line and run:

mechArm-M5 version:

```
# The default serial port name of the mechArm-M5 version is "/dev/ttyUSB0", and the baud rate is 115200". The serial port
roslaunch mecharm slider_control.py _port:=/dev/ttyUSB0 _baud:=115200
```

mechArm-PI version:

```
# The default serial port name of the mechArm-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".
roslaunch mecharm_pi slider_control.py _port:=/dev/ttyAMA0 _baud:=1000000
```

Note: Since the robot arm will move to the current position of the model when the command is input, make sure that the model in rviz does not appear to be worn out before you use the command.

Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm.

2 Model Following

In addition to the above controls, we can also let the model move by following the real robot arm. Open a command line and run:

mechArm-M5 version:

```
# The default serial port name of the mechArm-M5 version is "/dev/ttyUSB0", and the baud rate is 115200". The serial port
roslaunch mecharm follow_display.py _port:=/dev/ttyUSB0 _baud:=115200
```

mechArm-PI version:

```
# The default serial port name of the mechArm-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".
roslaunch mecharm_pi follow_display.py _port:=/dev/ttyAMA0 _baud:=1000000
```

Then open another command line and run:

mechArm-M5 version:

```
roslaunch mecharm follow.launch
```

mechArm-PI version:

```
roslaunch mecharm_pi follow.launch
```

It will **open rviz to show the model following effect**.

3 GUI control

On the basis of the previous contents, this package also **provides a simple GUI control interface**. This method is used for interaction between real robot arms. Connect to mecharm.

Open a command line:

mechArm-M5 version:

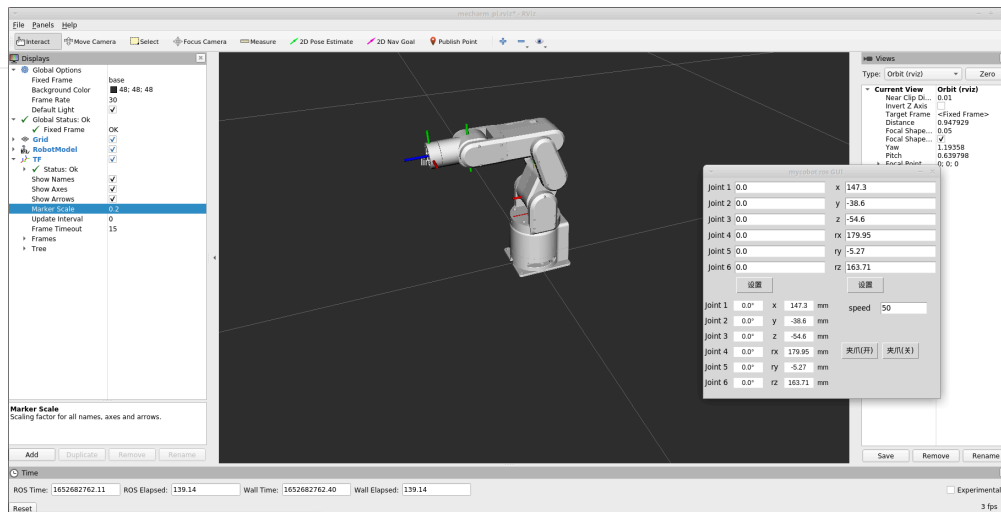
```
# The default serial port name of the mechArm-M5 version is "/dev/ttyUSB0", and the baud rate is 115200". The serial port
roslaunch mecharm simple_gui.launch port:=/dev/ttyUSB0 baud:=115200
```

mechArm-PI version:

```
# The default serial port name of the mechArm-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".
roslaunch mecharm_pi simple_gui.launch port:=/dev/ttyAMA0 baud:=1000000
```

Running effect:

1 myCobot Series 6-Axis Collaborative Robotic Arm



4 Keyboard control

Keyboard control is added in `mechArm` package, and real-time Synchronization is performed in rviz. This function depends on pythonApi, so be sure to connect with the real robot arm.

Open a command line and run:

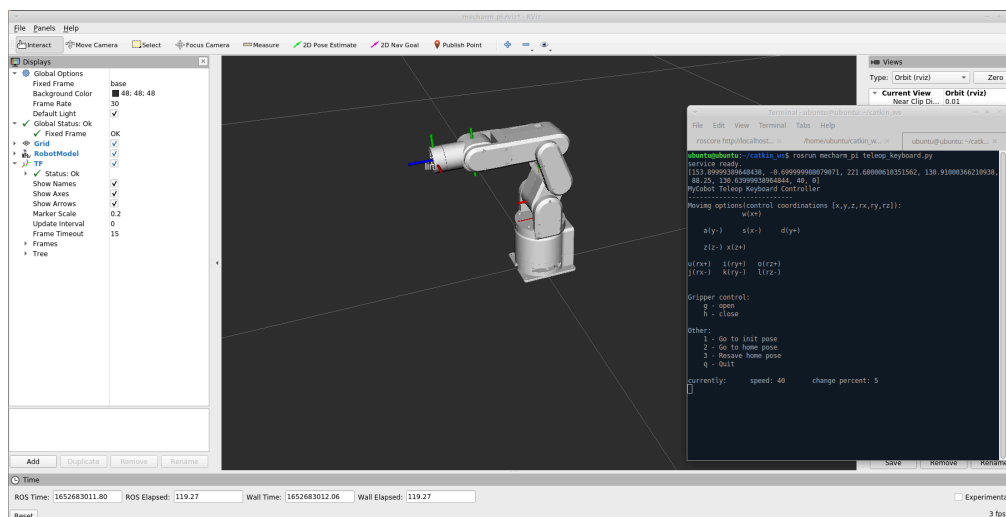
mechArm-M5 version:

```
# The default serial port name of the mechArm-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port  
roslaunch mecharm teleop_keyboard.launch port:=/dev/ttyUSB0 baud:=115200
```

mechArm-PI version:

```
# The default serial port name of the mechArm-PI version is "/dev/ttyAMA0", and the baud rate is 1000000.  
roslaunch mecharm_pi teleop_keyboard.launch port:=/dev/ttyAMA0 baud:=1000000
```

Running effect is as follows:



mechArm information will be output in the command line as follows:

SUMMARY

=====

PARAMETERS

```
* /mecharm_services/ baud: 1000000
* /mecharm_services/ port: /dev/ttyAMA0
* /robot_description: <?xml version="1....
* /rostdistro: melodic
* /rosversion: 1.14.11
```

NODES

```
/
  mecharm_services (mecharm_communication/mypal_services.py)
  real_listener (mecharm_pi/listen_real.py)
  robot_state_publisher (robot_state_publisher/robot_state_publisher)
  rviz (rviz/rviz)
```

```
ROS_MASTER_URI=http://localhost:11311
```

```
process[robot_state_publisher-1]: started with pid [14764]
process[rviz-2]: started with pid [14765]
process[mecharm_services-3]: started with pid [14766]
process[real_listener-4]: started with pid [14782]
[INFO] [1646649869.148017]: start ...
[INFO] [1646649869.156531]: /dev/ttyAMA0,1000000
```

Mycobot Status

Joint Limit:

```
joint 1: -160 ~ +160
joint 2: -90 ~ +90
joint 3: -180 ~ +45
joint 4: -160 ~ +160
joint 5: -100 ~ +100
joint 6: -180 ~ +180
```

```
Connect Status: True
```

```
Servo Infomation: unknown
```

```
Servo Temperature: unknown
```

```
Atom Version: unknown
```

```
[INFO] [1646649869.427899]: ready
```

Then open another command line and run:

mechArm-M5 version:

```
roslaunch mecharm teleop_keyboard.py
# or
roslaunch mecharm teleop_keyboard.py _speed:=70
```

mechArm-PI version:

```
roslaunch mecharm_pi teleop_keyboard.py
# or
roslaunch mecharm_pi teleop_keyboard.py _speed:=70
```

You will see the following output in the command line:

```
mecharm Teleop Keyboard Controller
-----
Moving options(control coordinations [x,y,z,rx,ry,rz]):
    w(x+)

    a(y-)    s(x-)    d(y+)

    z(z-)    x(z+)

    u(rx+)    i(ry+)    o(rz+)
    j(rx-)    k(ry-)    l(rz-)

Gripper control:
    g - open
    h - close

Other:
    1 - Go to init pose
    2 - Go to home pose
    3 - Resave home pose
    q - Quit

currently:    speed: 40    change percent: 5
```

Parameters supported by this script:

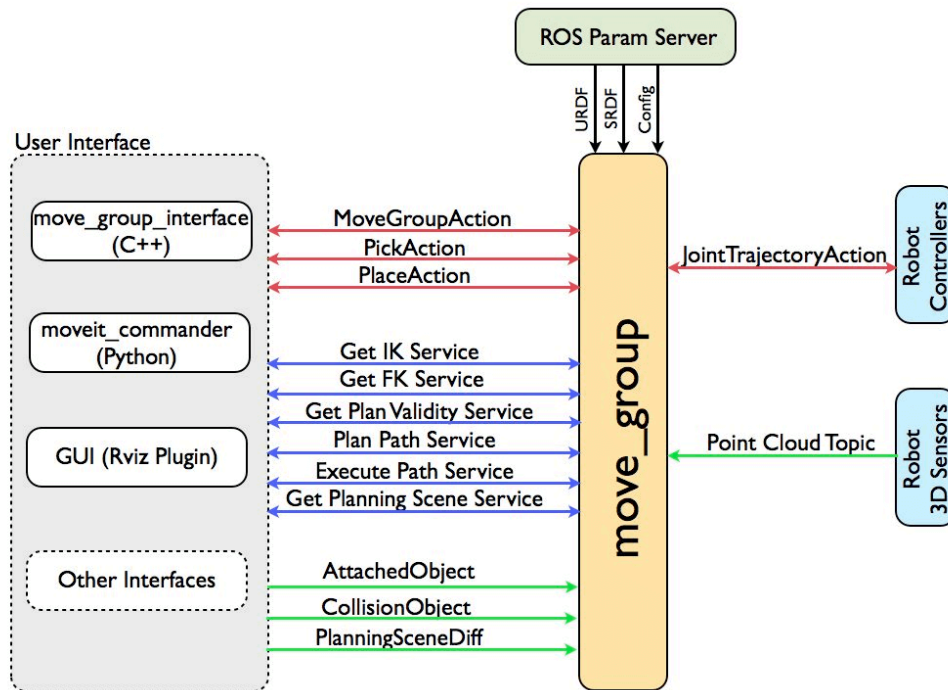
- `_speed`: the movement speed of the robot arm
- `_change_percent`: movement distance percentage

Movelt

1 Introduction to MoveIt

MoveIt is an integrated development platform in ROS, which consists of a variety of functional packages for manipulating robot arms, including motion planning, operation, control, inverse kinematics, 3D perception, collision detection, etc.

The following figure shows the high-level structure of the main node **move_group** provided by MoveIt. It is like a combiner: all the individual components are integrated together, providing a series of actions and services for users to use.



2 User interface

The user may access the operations and services provided by **move_group** in three ways:

- In C++, you may use **move_group** easily by using **move_group_interface** package.
- In Python, use the **moveit_commander** package.
- Via GUI: use Rviz (ROS visualization tool) of Motion-commander.

move_group can be configured using the ROS parameter server, from which the robot's URDF and SRDF can also be obtained.

3 Configuration

move_group is a ROS node. It uses the ROS parameter server to obtain three kinds of information:

- **URDF** - **move_group** looks for the **robot_description** parameter in the ROS parameter server to get the robot's URDF.
- **SRDF** - **move_group** looks for the **robot_description_semantic** parameter in the ROS parameter server to get the robot's SRDF. SRDF is typically created by the user using an MoveIt Setup Assistant.
- **MoveIt configuration** - **move_group** will look in the ROS parameter server for additional MoveIt-specific configurations, including joint constraint, kinematics, motion planning, perception, and other information. w The

configuration files for these components are automatically generated by the Movelt Setup Assistant and stored in the configuration directory of the robot's corresponding Movelt configuration package.

mechArm 270 Moveit

mycobot_ros has integrated the MoveIt section.

Open the command line and run:

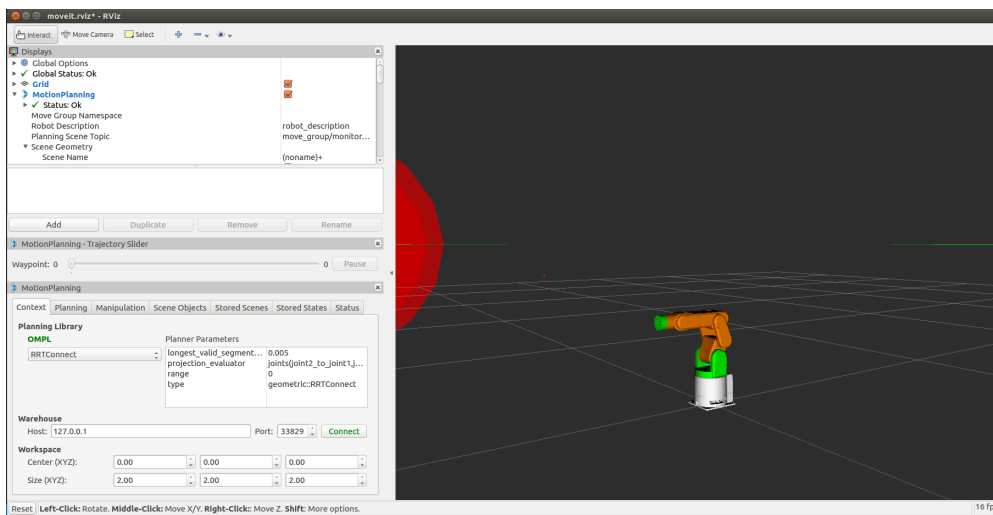
mechArm-M5 version:

```
roslaunch mecharm_moveit mecharm_moveit.launch
```

mechArm-PI version:

```
roslaunch mecharm_pi_moveit mecharm_moveit.launch
```

The operation effect is as follows:



If you want a real robot arm to execute a plan synchronously, you need to open another command line and run:

mechArm-M5 version:

```
# The default serial port name of the mechArm-M5 version is "/dev/ttyUSB0", and the baud rate is 115200". The serial port  
roslaunch mecharm_moveit sync_plan.py _port:=/dev/ttyUSB0 _baud:=115200
```

mechArm-PI version:

```
# The default serial port name of the mechArm version is "/dev/ttyAMA0", and the baud rate is 1000000".  
roslaunch mecharm_pi_moveit sync_plan.py _port:=/dev/ttyAMA0 _baud:=1000000
```


ROS2 introduction

The predecessor of ROS2 is ROS, and ROS is the Robot Operating System (Robot Operating System). But ROS itself is not an operating system, but a software library and toolset. The emergence of Ros solved the communication problem of each component of the robot. Later, more and more robot algorithms were integrated into ROS. ROS2 inherited ROS, which is more powerful and better than ROS.

1 Design goals and features of ROS2

ROS2 has the historical mission of changing the era of intelligent robots. At the beginning of the design, it was considered to meet the needs of various robot applications.

- **Multi-Robot Systems:** In the future, robots will not be independent individuals, and communication and collaboration between robots are also required. ROS2 provides standard methods and communication mechanisms for the application of multi-robot systems.
- **Cross-platform:** Robot application scenarios are different, and the control platforms used will also be very different. In order to allow all robots to run ROS2, ROS2 can run on Linux, Windows, MacOS, and RTOS across platforms.
- **Real time:** Robot motion control and many behavior strategies require the robot to be real-time. For example, the robot must reliably detect pedestrians in front of it within 100ms, or complete kinematics and dynamics calculations within 1ms. ROS2 is a real-time like this Basic requirements are provided.
- **Productization:** A large number of robots have entered our lives, and there will be more and more in the future, ROS2 can not only be used in the robot research and development stage, but also can be directly installed in the product and go to the consumer market. This also poses a huge challenge to the stability and robustness of ROS2.
- **Project management:** Robot development is a complex system engineering. The project management tools and mechanisms for the whole process of design, development, debugging, testing, and deployment will also be reflected in ROS2, making it easier for us to develop a robot.

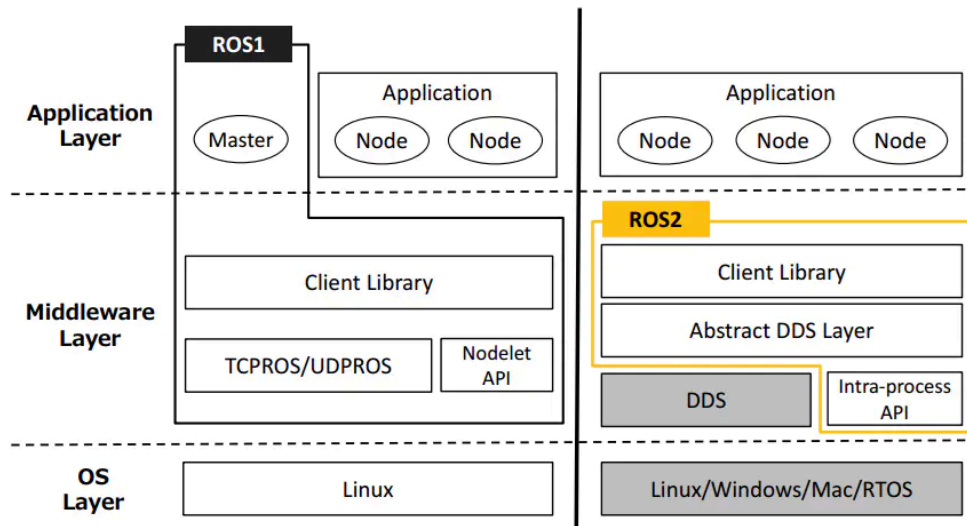
2 Release Version

The release version and maintenance cycle corresponding to ROS2 and Ubuntu.

ROS2 version	release date	Maintenance deadline	Ubuntu version
Dashing	2019.5	2021.5	Ubuntu 18.04 (Bionic Beaver)
Eloquent	2019.11	2020.11	Ubuntu 18.04 (Bionic Beaver)
Foxy	2020.6	2023.5	Ubuntu 20.04(Focal Fossa)
Galactic	2021.5	2022.11	Ubuntu 20.04(Focal Fossa)
Humble	2022.5	2027.5	Ubuntu 22.04(Jammy Jellyfish)

3 Comparison of ROS and ROS2

ROS2 redesigned the system architecture. The architecture changes between the two generations of ROS are as follows:



- **OS Layer:** In ROS2, it can be built on linux or other systems, even bare metal without an operating system.
- **Middleware Layer:** The communication system of ROS1 is based on TCPPROS/UDPPROS, while the communication system of ROS2 is based on DDS. DDS is a standard solution for data publishing/subscribing in distributed real-time systems.
- **Application Layer:** ROS1 relies on ROS Master, while in ROS2, a discovery mechanism called "Discovery" is used between nodes to help establish connections with each other.

ROS has designed a complete set of communication mechanisms (topics, services, parameters, actions) to simplify robot development. Through this mechanism, the various components of the robot can be connected. This mechanism has designed a node called Ros Master, and the communication of all other components must go through the master node. Once the master node hangs up, it will cause the communication of the entire robot system to collapse! Therefore, the instability of Ros cannot be used to make some high-risk robots such as automatic driving. In addition, there are the following disadvantages:

- Communication based on TCP has poor real-time performance and high system overhead
- Unfriendly to python3 support
- Messaging mechanism is not compatible
- No encryption mechanism, low security

ROS2 first removes the master node that exists in ROS. After removing the master node, each node can discover each other through the DDS node, each node is equal, and can realize one-to-one, one-to-many, and many-to-many communication. After using DDS for communication, reliability and stability have been enhanced.

Compared with **ROS** that only supports Linux systems, **ROS2** also supports windows, mac, and even RTOS platforms

Environment Installation

M5 Version:

Notice: When installing the virtual machine system, please install the **Ubuntu 20.04 version of the system**, The installation method is consistent with ubuntu 18.04.

To install different versions of ubuntu system in Linux, please refer to [13.1.1 Environment Building](#) chapter.

1 ROS Environment building

1.1 Installing ROS2

Building a basic development environment requires the installation of the robot operating system (short for ROS2) and a git version manager. The installation methods and procedures are described respectively below.

For **myCobot 280-M5**、**myCobot 320-M5 devices**、**myPalletizer 260-M5** and **mechArm 270-M5**, refer to the installation methods and procedures described below. For **myCobot 280-PI**、**myCobot 320-PI**、**myPalletizer 260-PI** and **mechArm 270-PI** devices, you only need to install the mycobot_ros installation package.

1.1.1 Selecting a version

There is a one-to-one relationship between ROS2 and ubuntu. Different versions of ubuntu correspond to different versions of ROS2. The reference website is as follows: <http://docs.ros.org/en/foxy/Releases.html>

Here are the ROS versions supported by Ubuntu:

ROS2 version	release date	Maintenance deadline	Ubuntu version
Foxy	June 5, 2020	May 2023	Ubuntu 20.04(Focal Fossa)
Galactic	May 23, 2021	November 2022	Ubuntu 20.04(Focal Fossa)
Humble	May 23, 2022	May 2027	Ubuntu 22.04(Jammy Jellyfish)

Install the ROS2 version corresponding to the Ubuntu version you have installed.

If the versions are different, the downloading will fail. The system we choose here is Ubuntu 20.04(recommend), so the corresponding ROS2 version is ROS2 Foxy.

NOTE: At present, we do not provide any reference for installing ROS in windows. If necessary, refer to <http://docs.ros.org/en/foxy/Installation/Alternatives/Windows-Development-Setup.html>

1.1.2 Installing ROS

1 Adding a software source

There is no ROS2 software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS2. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

- official source:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org
```

- If the download speed is slow, it is recommended to select a nearby mirror source to replace the above command. For example, huawei cloud is:

```
echo "deb [arch=$(dpkg --print-architecture)] https://repo.huaweicloud.com/ros2/ubuntu/ $(lsb_release -cs) main" | sudo te
```

2 Setting a key

Configuring a public key. This step is to let the system confirm that our path is safe, so that there is no problem in downloading the file. Otherwise it will be deleted immediately after downloading:

```
sudo apt install curl gnupg2 -y  
curl -s https://gitee.com/ohhuo/rosdistro/raw/master/ros.asc | sudo apt-key add -
```

3 Installation

After adding a new software source, you need to update the software source list. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command:

```
sudo apt-get update
```

Execute **Installing ROS2**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command according to your Ubuntu version:

```
# Ubuntu 20.04 foxy version  
sudo apt install ros-foxy-desktop
```

```
# Ubuntu 20.04 galactic version  
sudo apt install ros-galactic-desktop
```

```
# Ubuntu 22.04 humble version  
sudo apt install ros-humble-desktop
```

The installation process takes a long time, so please wait with patience.

Refresh environment variables after installation is complete:

```
source /opt/ros/foxy/setup.bash
```

1.1.3 Set up the ros2 environment

In order to avoid re-validating the ROS2 function path every time the terminal window is closed, we can **set the path to an environment variable**, so that each time you open a new terminal, the ROS2 function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

1 myCobot Series 6-Axis Collaborative Robotic Arm

Execute the following commands:

```
# Ubuntu 20.04 foxy version
# Add the ros2 environment to the environment variables of the current console
echo "source /opt/ros/foxy/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04 galactic version
echo "source /opt/ros/galactic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 22.04 humble version
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

Installing a ROS2 extra dependency

Input the following command in the terminal to install a ROS2 extra dependency, and open a console terminal (shortcut key: Ctrl+Alt+T):

```
sudo apt install python3-argcomplete -y
```

```
sudo apt install ros-foxy-xacro
```

```
sudo apt-get install python3-colcon-common-extensions
```

```
# Ubuntu 20.04 foxy version
sudo apt install ros-foxy-joint-state-publisher-gui
```

```
# Ubuntu 20.04 galactic version
sudo apt install ros-galactic-joint-state-publisher-gui
```

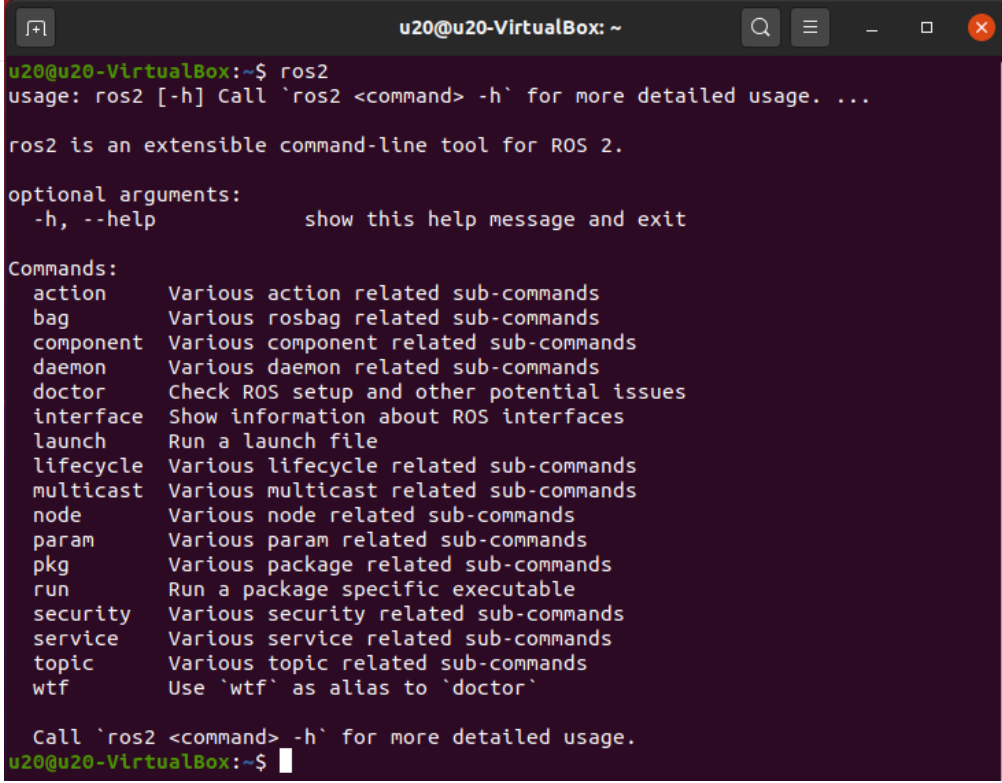
```
# Ubuntu 22.04 humble version
sudo apt install ros-humble-joint-state-publisher-gui
```

1.1.4 Verify installation

To verify whether ROS2 has been installed successfully, open a console terminal (shortcut key: Ctrl+Alt+T) and execute the following command in the terminal:

```
ros2
```

When the following interface is displayed, it means that ROS2 has been installed successfully.



```

u20@u20-VirtualBox:~$ ros2
usage: ros2 [-h] Call 'ros2 <command> -h' for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit

Commands:
  action      Various action related sub-commands
  bag         Various rosbag related sub-commands
  component   Various component related sub-commands
  daemon      Various daemon related sub-commands
  doctor       Check ROS setup and other potential issues
  interface    Show information about ROS interfaces
  launch      Run a launch file
  lifecycle    Various lifecycle related sub-commands
  multicast    Various multicast related sub-commands
  node         Various node related sub-commands
  param        Various param related sub-commands
  pkg          Various package related sub-commands
  run          Run a package specific executable
  security     Various security related sub-commands
  service      Various service related sub-commands
  topic        Various topic related sub-commands
  wtf          Use 'wtf' as alias to 'doctor'

Call 'ros2 <command> -h' for more detailed usage.
u20@u20-VirtualBox:~$

```

2 git Installation

2.1 Updating the software source list

Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window to **update the software source list**:

```
sudo apt-get update
```

2.2 Installing git

Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window to **execute the installation of git**:

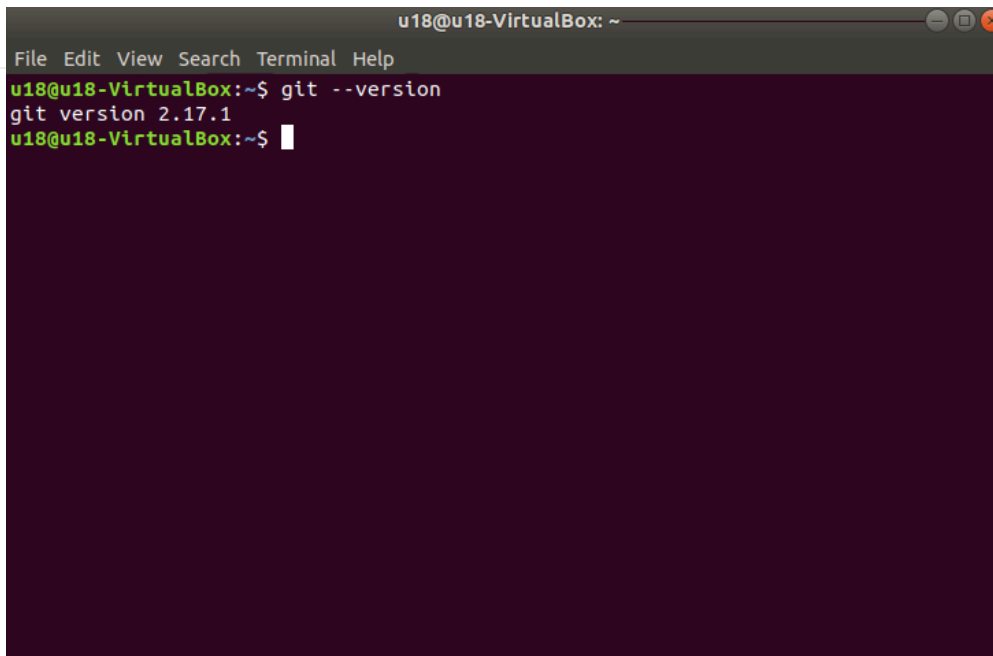
```
sudo apt-get install git
```

2.3 Verify installation

Read the git version, open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command on the terminal window:

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, the installation is successful.



```
u18@u18-VirtualBox: ~  
File Edit View Search Terminal Help  
u18@u18-VirtualBox:~$ git --version  
git version 2.17.1  
u18@u18-VirtualBox:~$
```

3 mycobot_ros2 Installation

`mycobot_ros2` is a ROS2 package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros2

3.1 Precondition

Before installing the package, make sure you have a ros2 workspace.

Here is an **example command for creating a workspace**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command in the command line:

```
# Create a folder  
mkdir -p ~/colcon_ws/src  
  
# compile workspace  
cd ~/colcon_ws  
colcon build
```

Add workspace environment

The official default ROS2 workspace is `colcon_ws`.

```
echo "source ~/colcon_ws/install/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

3.2 Installation

NOTE:

- This package depends on ROS2 and MoveIt2. Before using it, make sure that you have installed ROS2 and MoveIt2 successfully.
- The interaction of this package with a real robot arm depends on PythonApi - `pymycobot` .
- The Api project location is: <https://github.com/elephantrobotics/pymycobot>
- Quick installation: `pip install pymycobot --upgrade`

When executing the `pip install pymycobot --upgrade` command, if the following error message appears:

```
u182@u182-VirtualBox:~$ pip install pymycobot --upgrade
Command 'pip' not found, but can be installed with:
sudo apt install python-pip
```

Enter the following command to install pip at the prompt

```
sudo apt install python-pip
```

- If your Ubuntu system is version 20.04, please execute the command `sudo apt install python3-pip` to install pip
- After the pip installation is complete, the terminal executes again

```
pip install pymycobot --upgrade
```

- The installation method depends on Git, so make sure that Git have been installed in your computer.

The official default ROS2 workspace is `colcon_ws` .

```
cd ~/colcon_ws/src # Enter the src folder of the workspace
# Clone the code on github
git clone --depth 1 https://github.com/elephantrobotics/mycobot_ros2.git
cd .. # return to the workspace
colcon build --symlink-install # Build the code in the workspace, --symlink-install: Avoid having to recompile python scri
source install/setup.bash # add environment variables
```

PI Version

The Raspberry Pi version comes with an Ubuntu (V-20.04) system and a built-in development environment, so you don't need to build and manage it, Just update the `mycobot_ros2` package.

`mycobot_ros2` is a ROS2 package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros2

Robotic arm API driver library address: <https://github.com/elephantrobotics/pymycobot>

1 Update the mycobot_ros2 package

In order to ensure that users can use the latest official package in time, they can go to the `/home/er/colcon_ws/src` folder through the file manager and open the console terminal (shortcut `Ctrl+Alt+T`), enter the following command to update:


```
# Clone the code on github
cd ~/colcon_ws/src
git clone https://github.com/elephantrobotics/mycobot_ros2.git # Please check the attention section below before deciding
cd .. # Back to work area
colcon build --symlink-install # Build the code in the workspace, --symlink-install: Avoid having to recompile python scri
source install/setup.bash # add environment variables
```

Note: If the `mycobot_ros2` folder already exists in the `/home/er/colcon_ws/src` (equivalent to `~/colcon_ws/src`) directory, you need to delete the original `mycobot_ros2` before executing the above command. Among them, `er` in the directory path is the user name of the virtual machine. If it is inconsistent, please modify it.

So far, the ROS2 environment construction has been completed. Please refer to [13.2.3 Rviz Introduction and Use](#) for the use of ROS2.

Basic tool

In this chapter, you will learn about the common command tools of ROS2.

1 Topics

ROS 2 breaks complex systems down into many modular nodes. Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages. Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.

Specific reference: [Official Tutorials](#)

- **topics help**

```
ros2 topics -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node  
ros2 run turtlesim turtle_teleop_key
```

- **Node Relationship Diagram**

```
rqt_graph
```

- **Learn about topic-related commands**

```
ros2 topics -h
```

- **topics list**

```
ros2 topic list  
ros2 topic list -t # Display the corresponding message type
```

- **View topic content**

```
ros2 topic echo <topic_name>  
ros2 topic echo /turtle1/cmd_vel
```

- **Display topic-related information, type**

```
ros2 topic info <topic_name>  
# Output /turtle1/cmd_vel topic related information  
ros2 topic info /turtle1/cmd_vel
```

- **Display interface related information**

```
ros2 interface show <msg_type>

# Output geometry_msgs/msg/Twist interface related information

ros2 interface show geometry_msgs/msg/Twist
```

- **Issue an order**

```
ros2 topic pub <topic_name> <msg_type> '<args>'

# Issue speed command

ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"

# Issue speed commands at a certain frequency

ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```

- **See how often topics are posted**

```
ros2 topic hz <topic_name>

# Output /turtle1/cmd_vel publish frequency

ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```

2 Nodes

Each node in ROS should be responsible for a single, module purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc). Each node can send and receive data to other nodes via topics, services, actions, or parameters. A full robotic system is comprised of many nodes working in concert. In ROS 2, a single executable (C++ program, Python program, etc.) can contain one or more nodes.

Specific reference: [Official Tutorials](#)

- **nodes help**

```
ros2 nodes -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node

ros2 run turtlesim turtle_teleop_key
```

- **View the node list**

```
ros2 node list
```

- **View Node Relationship Diagram**

```
rqt_graph
```

- **Remapping**

```
ros2 run turtlesim turtlesim_node --ros-args --remap __node:=my_turtle
ros2 node list
```

- **View node information**

```
ros2 node info <node_name>
ros2 node info /my_turtle
```

3 Services

Services are another method of communication for nodes in the ROS graph. Services are based on a call-and-response model, versus topics' publisher-subscriber model. While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.

Specific reference: [Official Tutorials](#)

- **services help**

```
ros2 service -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View the service list**

```
ros2 service list
# Display service list and message type
ros2 service list -t
```

- **View the message types received by the service**

```
ros2 service type <service_name>
ros2 service type /clear
```

- **Find services that use a certain message type**

```
ros2 service find <type_name>
ros2 service find std_srvs/srv/Empty
```

- **View Service Message Type Definitions**

```
ros2 interface show <type_name>.srv
ros2 interface show std_srvs/srv/Empty.srv
```

- **Call the service command to clear the walking track**

```
ros2 service call <service_name> <service_type>
ros2 service call /clear std_srvs/srv/Empty
```

- **Spawn a new turtle**

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: 'turtle2'}"
```

4 Parameters

A parameter is a configuration value of a node. You can think of parameters as node settings. A node can store parameters as integers, floats, booleans, strings, and lists. In ROS 2, each node maintains its own parameters. For more background on parameters, please see the concept document.

Specific reference: [Official Tutorials](#)

- **parameters help**

```
ros2 param -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View service list**

```
ros2 param list
```

- **Get the parameter value**

```
ros2 param get <node_name> <parameter_name>
ros2 param get /turtlesim background_g
```

- **Set parameter values**

```
ros2 param set <node_name> <parameter_name> <value>
ros2 param set /turtlesim background_r 150
```

- **Export parameter values**

```
ros2 param dump <node_name>
ros2 param dump /turtlesim
```

- **Import parameters independently**

```
ros2 param load <node_name> <parameter_file>
ros2 param load /turtlesim ./turtlesim.yaml
```

- **Start the node and import parameters at the same time**

```
ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>
ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```

5 Actions

Actions are one of the communication types in ROS 2 and are intended for long running tasks. They consist of three parts: a goal, feedback, and a result.

Actions are built on topics and services. Their functionality is similar to services, except actions are preemptable (you can cancel them while executing). They also provide steady feedback, as opposed to services which return a single response.

Actions use a client-server model, similar to the publisher-subscriber model (described in the topics tutorial). An “action client” node sends a goal to an “action server” node that acknowledges the goal and returns a stream of feedback and a result.

Specific reference: [Official Tutorials](#)

- **action help**

```
ros2 action -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

Press G|B|V|C|D|E|R|T to achieve rotation, press F to cancel

- **View the server and client of the node action**

```
ros2 node info /turtlesim
```

- **View action list**

```
ros2 action list
ros2 action list -t # show action type
```

- **view action info**

```
ros2 action info <action>
ros2 action info /turtle1/rotate_absolute
```

- **View action message content**

```
ros2 interface show turtlesim/action/RotateAbsolute
```

- **Send action target information**

```
ros2 action send_goal <action_name> <action_type>
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"
# With feedback information
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 0}" --feedback
```

6 RQt

RQt is a graphical user interface framework that implements various tools and interfaces in the form of plugins. One can run all the existing GUI tools as dockable windows within RQt! The tools can still run in a traditional standalone method, but RQt makes it easier to manage all the various windows in a single screen layout.

Specific reference: [Official Tutorials](#)

You can run any RQt tools/plugins easily by:

```
rqt
```

- **rqt help**

```
rqt -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- Action Type Browser: / Plugins -> Actions -> Action Type Browser
- parameter reconfiguration: / Plugins -> configuration -> Parameter Reconfigure
- Node grap: /Node Graph
- control steering: /Plugins -> Robot Tools -> Robot Steering
- service invocation: /Plugins -> Services -> Service Caller
- Service Type Browser: Plugins -> Services -> Service Type Browser
- message release: Plugins -> Topics -> Message Publisher
- Message Type Browser: Plugins -> Topics -> Message Type Browser
- topic list: Plugins -> Topics -> Topic Monitor
- draw a graph: Plugins -> Visualization -> Plot
- **View logs: rqt_console**

```
ros2 run rqt_console rqt_console
ros2 run turtlesim turtlesim_node
ros2 topic pub -r 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z
```

7 TF2

tf2 is the transform library, which lets the user keep track of multiple coordinate frames over time. tf2 maintains the relationship between coordinate frames in a tree structure buffered in time and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time.

Specific reference: [Official Tutorials](#)

Let's start by installing the demo package and its dependencies.

```
sudo apt-get install ros-foxy-turtle-tf2-py ros-foxy-tf2-tools ros-foxy-tf-transformations
```

- **follow**
- launch starts 2 little turtles, the first little turtle automatically follows the second one

```
ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py
```

- Control the movement of the first little turtle through the keyboard

```
ros2 run turtlesim turtle_teleop_key
```

- **View TF tree**

```
ros2 run tf2_tools view_frames.py
evince frames.pdf
```

- **View the relationship between two coordinate systems**

```
ros2 run tf2_ros tf2_echo [reference_frame] [target_frame]
ros2 run tf2_ros tf2_echo turtle2 turtle1
```

- **View TF relationships on rviz**

```
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix --share turtle_tf2_py)/rviz/turtle_rviz.rviz
```

8 URDF

URDF is the Unified Robot Description Format for specifying robot geometry and organization in ROS.

Specific reference: [Official Tutorials](#)

- **Complete syntax**


```

<robot>

  # describe:

  # Parameters: name=""

  # Child node:

    <link>

      # Description:

      # Parameters: name=""

      # Child node:

        <visual>

          # describe:

          # Parameters:

          # child nodes:

            <geometry>

              # description

              # parameters

              # Child node:

                <cylinder />

                  # Description:

                  # Parameters:

                    # length="0.6"

                    # radius="0.2"

                <box />

                  # description

                  # Parameters:size="0.6 0.1 0.2"

                <mesh />

                  # Description

                  #Parameters: filename="package://urdf_tutorial/meshes/l_finger_tip.dae"

            <collision>

              # Description: collision element, prioritized

              # parameters

              # child node

                <geometry>

            <inertial>

              # description

              # parameters

              # Child nodes:

                <mass />

                  # description: mass

                  # Parameters: value=10

                <inertia />

                  # Description: Inertia

                  # Parameters: i+"Cartesian product of xyz" (9 in total)="0.4"

            <origin />

              # Description:

              # Parameters:

                # rpy="0 1.5 0"

                # xyz="0 0 -0.3"

            <material />

              # Description

              # Parameters: name="blue"

```

```

<joint>
  # Description
  # Parameters:
    # name=""
    # type=""
    # fixed
    # prismatic
  # child node
  <parent />
    # Description
    # Parameters: link=""
  <child />
    # Description:
    # Parameters: link=""

  <origin />
    # Description:
    # Parameters: xyz="0 -0.2 0.25"

  <limit />
    # Description
    # Parameters:
      # effort="1000.0"    maximum effort
      # lower="-0.38"     Joint upper limit (radians)
      # upper="0"         Joint lower limit (radians)
      # velocity="0.5"    Maximum velocity

  <axis />
    # Description: Press ? axis rotation
    # Parameters: xyz="0 0 1", along the Z axis

<material>
  # Description:
  # Parameters: name="blue"
  # child node:
    <color />
      # description:
      # Parameters: rgba="0 0 0.8 1"

```

- **Install dependent libraries**

```

sudo apt install ros-foxy-joint-state-publisher-gui ros-foxy-joint-state-publisher
sudo apt install ros-foxy-xacro

```

- **Download the source code**

```

cd ~/dev_ws
git clone -b ros2 https://github.com/ros/urdf_tutorial.git src/urdf_tutorial

```

- **Compiling the source code**

```
colcon build --packages-select urdf_tutorial
```

- **Running the example**

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/01-myfirst.urdf
```

9 Launch

The launch system in ROS 2 is responsible for helping the user describe the configuration of their system and then execute it as described. The configuration of the system includes what programs to run, where to run them, what arguments to pass them, and ROS-specific conventions which make it easy to reuse components throughout the system by giving them each a different configuration. It is also responsible for monitoring the state of the processes launched, and reporting and/or reacting to changes in the state of those processes.

Launch files written in Python, XML, or YAML can start and stop different nodes as well as trigger and act on various events.

Specific reference: [Official Tutorials](#)

Setup

Create a new directory to store your launch files:

```
mkdir launch
```

Writer the launch file

Let's put together a ROS 2 launch file using the turtlesim package and its executables. As mentioned above.

Copy and paste the complete code into the launch/turtlesim_mimic_launch.py file:

```

from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            executable='mimic',
            name='mimic',
            remappings=[
                ('/input/pose', '/turtlesim1/turtle1/pose'),
                ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel'),
            ]
        )
    ])

```

Run the ros2 launch file

To run the launch file created above, enter into the directory you created earlier and run the following command:

The syntax format is:

```
ros2 launch <package_name> <launch_file_name>
```

```

cd launch
ros2 launch turtlesim_mimic_launch.py

```

- **launch help**

```
ros2 launch -h
```

- **running node**

```
ros2 launch turtlesim multisim.launch.py
```

- **Check the parameters of the launc file**

```
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py -s
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py --show-arguments
ros2 launch turtlebot3_bringup robot.launch.launch.py -s
```

- **Run the launch file with parameters**

```
ros2 launch turtlebot3_bringup robot.launch.launch.py usb_port:=/dev/opencr
```

- **Run the node and debug**

```
ros2 launch turtlesim turtlesim_node.launch.py -d
```

- **Only output node description**

```
ros2 launch turtlesim turtlesim_node.launch.py -p
```

- **running components**

```
ros2 launch composition composition_demo.launch.py
```

10 Run

run is used to run a single node, component program

- **run help**

```
ros2 run -h
```

- **running node**

```
ros2 run turtlesim turtlesim_node
```

- **Run node with parameters**

```
ros2 run turtlesim turtlesim_node --ros-args -r __node:=turtle2 -r __ns:=/ns2
```

- **Run component container**

```
ros2 run rclcpp_components component_container
```

- **running components**

```
ros2 run composition manual_composition
```

11 Package

A package can be considered a container for your ROS 2 code. If you want to be able to install your code or share it with others, then you'll need it organized in a package. With packages, you can release your ROS 2 work and allow others to build and use it easily.

Package creation in ROS 2 uses ament as its build system and colcon as its build tool. You can create a package using either CMake or Python, which are officially supported, though other build types do exist.

Specific reference: [Official Tutorials](#)

Creating a workspace

Create a new directory for every new workspace. The name doesn't matter, but it is helpful to have it indicate the purpose of the workspace. Let's choose the directory name ros2_ws, for "development workspace":

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
```

- **pkg help**

```
ros2 pkg -h
```

- **List Feature Packs**

```
ros2 pkg executable turtlesim
```

- **Output a function package executable program**

```
ros2 pkg executable turtlesim
```

- **Create a Python package**

Make sure you are in the src folder before running the package creation command.

```
cd ~/ros2_ws/src
```

The command syntax for creating a new package in ROS 2 is:

```
ros2 pkg create --build-type ament_python <package_name>
# you will use the optional argument --node-name which creates a simple Hello World type executable in the package.
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

- **Build a package**

Putting packages in a workspace is especially valuable because you can build many packages at once by running colcon build in the workspace root. Otherwise, you would have to build each package individually.

```
# Return to the root of your workspace:  
  
cd ~/ros2_ws  
  
# Now you can build your packages:  
  
colcon build
```

- **Source the setup file**

To use your new package and executable, first open a new terminal and source your main ROS 2 installation.

Then, from inside the `ros2_ws` directory, run the following command to source your workspace:

```
source install/setup.bash
```

Now that your workspace has been added to your path, you will be able to use your new package's executables.

- **Use the package**

To run the executable you created using the `--node-name` argument during package creation, enter the command:

```
ros2 run my_package my_node
```

Brief introduction and use of rviz2

rviz is a three-dimensional visualization platform in ROS. On the one hand, it can realize the graphical display of external information, and on the other hand, it can also release control information to the object through rviz, so as to realize the monitoring and control of the robot.

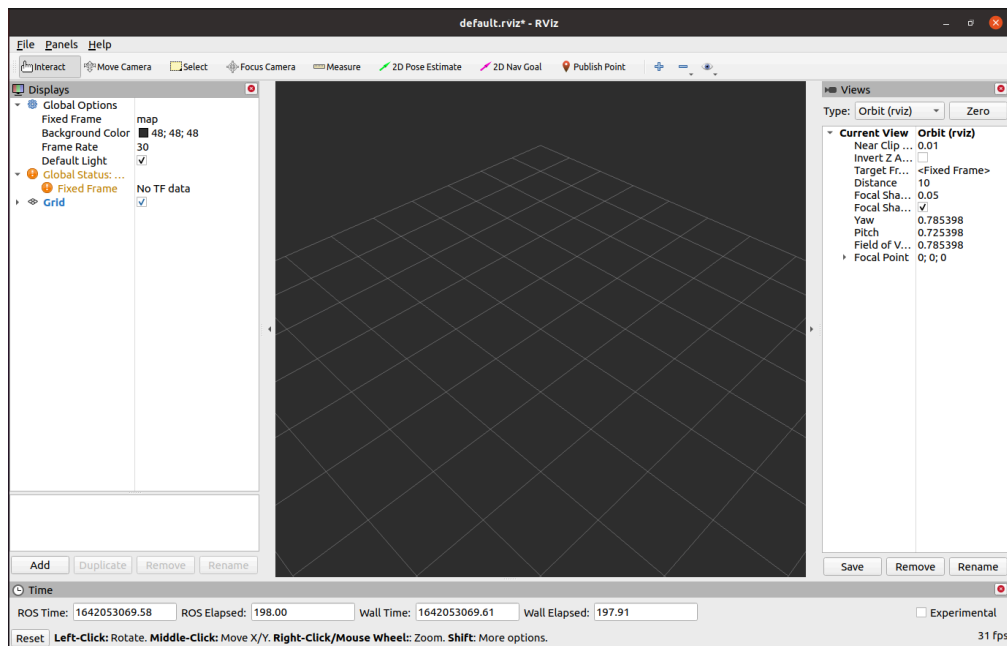
1 Introduction to rviz2

The successful installation of ros2 indicates that rviz2 is also successfully installed together, because the installation of ros2 includes rviz2.

Open a new terminal (shortcut `Ctrl+Alt+T`) enter the command to open rviz2

```
rviz2
```

Open rviz2 and display the following interface:



2 Introduction of each area

- On the left is the list of monitors, a monitor is something that draws something in the 3D world and may have some options available in the display list.
- Above is the toolbar, which allows users to use various function buttons to select tools with multiple functions
- The middle part is the 3D view: it is the main screen where various data can be viewed in 3D. The background color, fixed frame, grid, etc. of the 3D view can be set in detail in the Global Options and Grid items displayed on the left.
- Below is the time display area, including system time and ROS time.
- The right side is the observation angle setting area, and different observation angles can be set.

We only give a rough introduction in this part. If you want to know more detailed content, you can go to the [user guide](#) to view it.

3 mycobot_ros2 installation and update

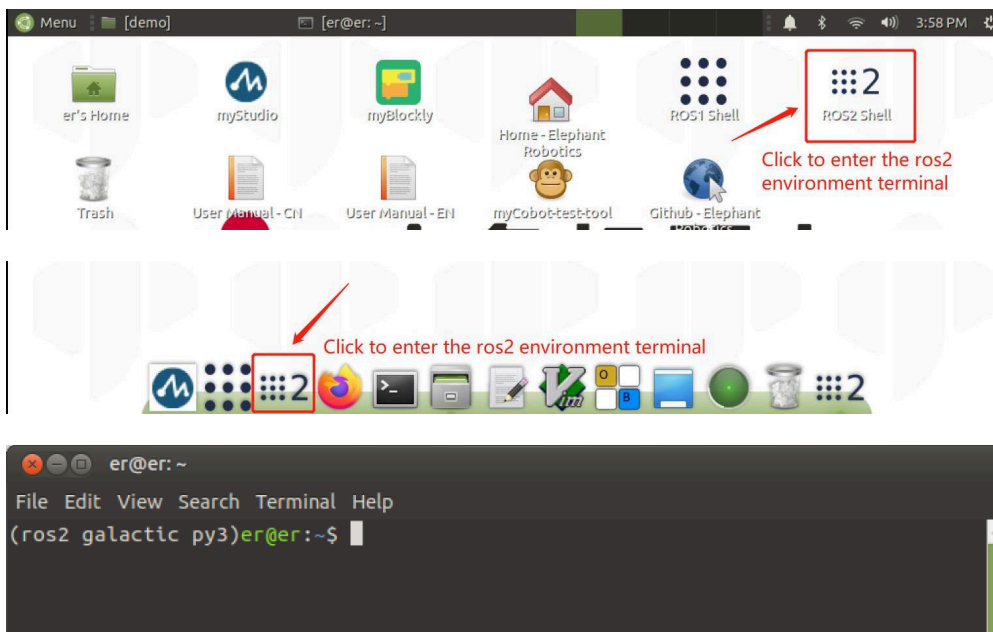
- **M5 version:** Please refer to the end of the [13.2.1 Installation of ROS2](#) chapter.
- **PI版本(Ubuntu 20.04):**

`mycobot_ros2` is a ROS2 package from ElephantRobotics that works with all types of desktop robots.

The address of the project: https://github.com/elephantrobotics/mycobot_ros2

The official default ROS2 workspace is `colcon_ws`.

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower bar of the desktop to open the ROS2 environment terminal:



Then enter the following command:

```
cd ~/colcon_ws/src # Enter the src folder of the workspace
# Clone the code on github
git clone https://github.com/elephantrobotics/mycobot_ros2.git
cd .. # return to the workspace
colcon build --symlink-install # Build the code in the workspace, --symlink-install: Avoid having to recompile python scri
source install/setup.bash # add environment variables
```

Note: If the `/home/er/colcon/src`(equivalent to `~/colcon_ws/src`) already exists in the `mycobot_ros2` folder, you need to delete the `mycobot_ros2` folder before running the above command.

4 Simple use

Launch via launch.py file

This example is based on the fact that you have completed the [environment setup](#) and successfully copied the company's code from GitHub.。

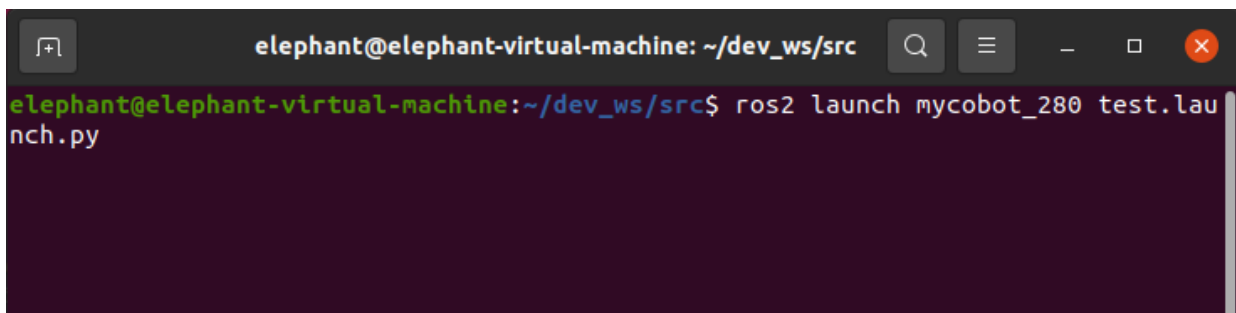
Open a console terminal (shortcut `Ctrl+Alt+T`) Enter the following command to **configure the ROS2 environment**.

```
cd ~/colcon_ws
colcon build --symlink-install
source install/setup.bash
```

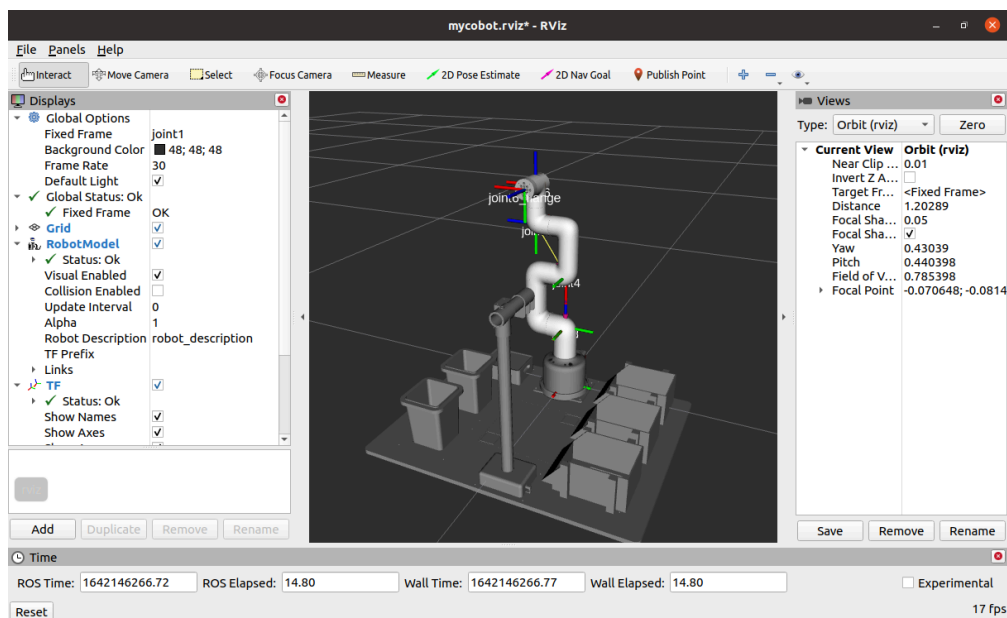
Enter again:

- mycobot 280-M5 version:

```
ros2 launch mycobot_280 test.launch.py
```



Open rviz and get the following result:



If you want to know more information about rviz, you can go to the [official documentation](#) to view it

5 M5 Version Prerequisites

- Type `Ctrl+Shift+T` in the command terminal to open another terminal window in the same directory to view the device name:

1 myCobot Series 6-Axis Collaborative Robotic Arm

```
# View the device name of the robotic arm
ls /dev/ttyUSB* # old version myCobot280 M5

# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # new version myCobot280 M5
```

- Grant the serial port permission to the robotic arm:

```
# The default device name is /dev/ttyUSB0, if the device name is not the default value, it needs to be modified.
sudo chmod 777 /dev/ttyUSB0 # old version myCobot280 M5

sudo chmod 777 /dev/ttyACM0 # new version myCobot280 M5
```

Then enter the user password(**Note:** The password is not displayed, just enter it correctly).

Control and following of the robot arm

1 Slider Control

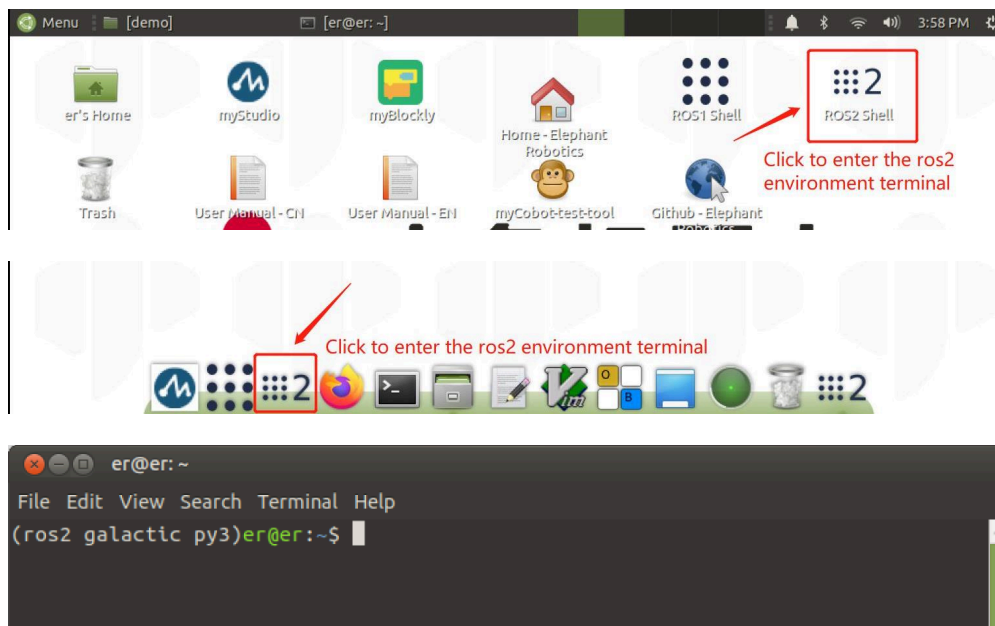
Open a command line and run:

- mecharm 270-M5 version:

```
# The default serial port name of the mechArm 270-M5 version is "/dev/ttyUSB0", and the baud rate is 115200".  
ros2 launch mecharm slider_control.launch.py
```

- mecharm 270-PI version:

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:

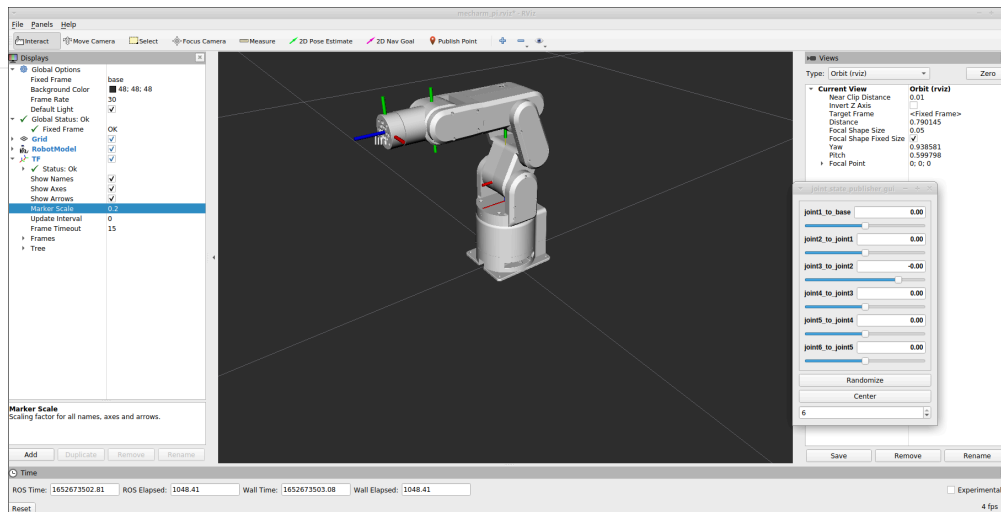


Then run the command:

```
# The default serial port name of the mechArm 270-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".  
ros2 launch mecharm_pi slider_control.launch.py
```

rviz and a slider component will be opened, and you will see the following interface:

1 myCobot Series 6-Axis Collaborative Robotic Arm



Then you can **control the model in rviz to make it move by dragging the slider**. The real mechArm will move with it.

Note: Since the robot arm will move to the current position of the model when the command is input, make sure that the model in rviz does not appear to be worn out before you use the command.

Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm.

2 Model Following

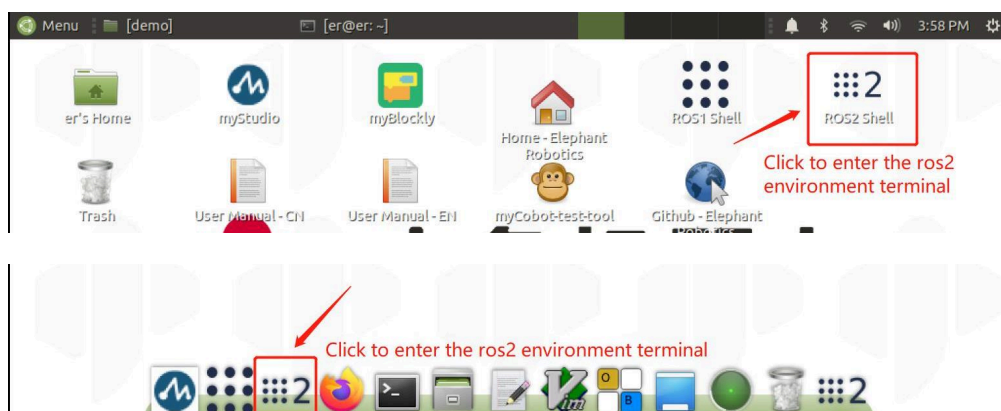
In addition to the above controls, we can also let the model move by following the real robot arm. Open a command line and run:

- mecharm 270-M5 version:

```
# The default serial port name of the mechArm version is "/dev/ttyUSB0", and the baud rate is 115200".  
ros2 launch mecharm mycobot_follow.launch.py
```

- mecharm 270-PI version:

Click the **ROS2 Shell** icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



```
er@er: ~  
File Edit View Search Terminal Help  
(ros2 galactic py3)er@er:~$
```

Then run the command:

```
# The default serial port name of the mechArm 270-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".  
ros2 launch mecharm_pi mycobot_follow.launch.py
```

It will **open rviz to show the model following effect**.

3 GUI control

On the basis of the previous contents, this package also **provides a simple GUI control interface**. This method is used for interaction between real robot arms. Connect to mecharm.

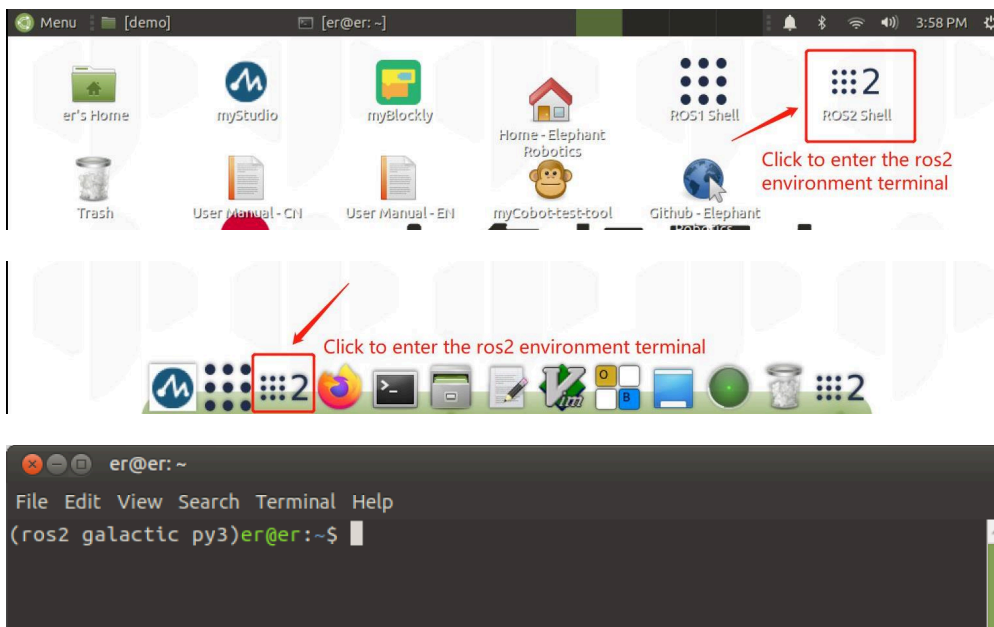
- mecharm 270-M5 version:

Open a command line:

```
# The default serial port name of the mechArm 270-M5 version is "/dev/ttyUSB0", and the baud rate is 115200".  
ros2 launch mecharm simple_gui.launch.py
```

- mecharm 270-PI version:

Click the **ROS2 Shell** icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:

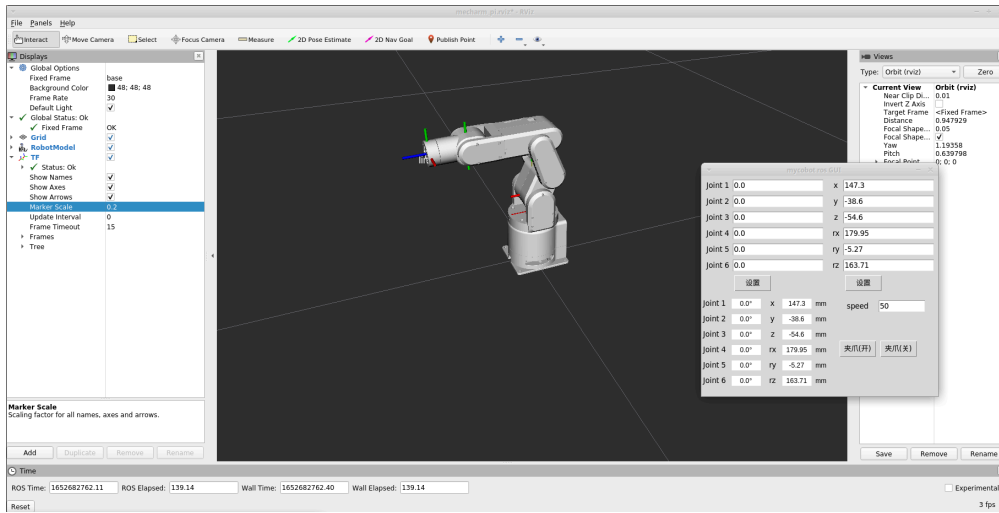


Then run the command:

1 myCobot Series 6-Axis Collaborative Robotic Arm

```
# The default serial port name of the mechArm 270-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".  
ros2 launch mecharm_pi simple_gui.launch.py
```

Running effect:



4 Keyboard control

Keyboard control is added in `mecharm` package, and real-time Synchronization is performed in rviz. This function depends on pythonApi, so be sure to connect with the real robot arm.

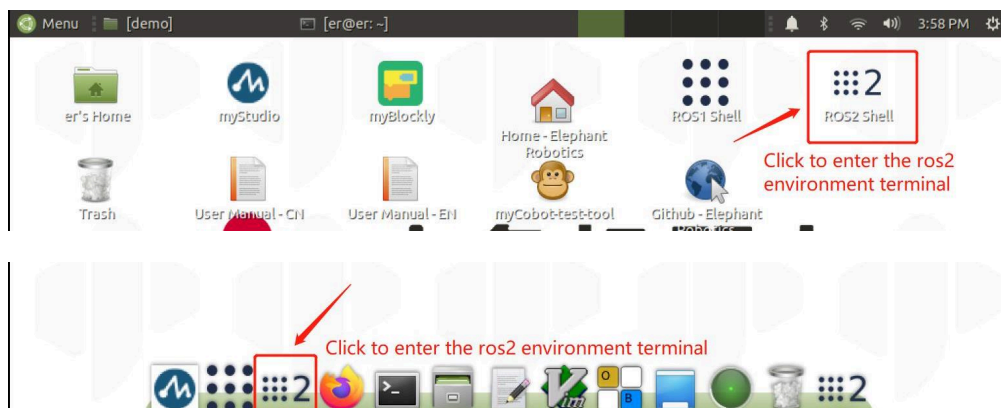
Open a command line and run:

- mecharm 270-M5 version:

```
# The default serial port name of the mechArm 270-PI version is "/dev/ttyUSB0", and the baud rate is 115200".  
ros2 launch mecharm teleop_keyboard.launch.py
```

- mecharm 270-PI version:

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



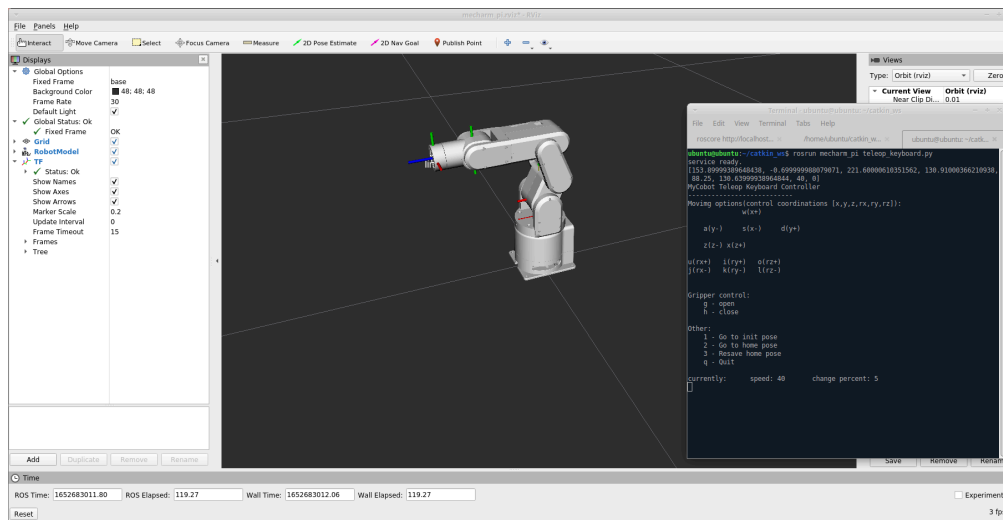
1 myCobot Series 6-Axis Collaborative Robotic Arm

```
er@er: ~  
File Edit View Search Terminal Help  
(ros2 galactic py3)er@er:~$
```

Then run the command:

```
# The default serial port name of the mechArm 270-PI version is "/dev/ttyAMA0", and the baud rate is 1000000".  
ros2 launch mecharm_pi teleop_keyboard.launch.py
```

Running effect is as follows:



mechArm information will be output in the command line as follows:


```
[INFO] [launch]: All log files can be found below /home/u20/.ros/log/2022-08-01-14-53-35-880311-u20-VirtualBox-5591
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [robot_state_publisher-1]: process started with pid [5594]
[INFO] [rviz2-2]: process started with pid [5596]
[INFO] [follow_display-3]: process started with pid [5598]
[robot_state_publisher-1] Parsing robot urdf xml string.
[robot_state_publisher-1] Link link1 had 1 children
[robot_state_publisher-1] Link link2 had 1 children
[robot_state_publisher-1] Link link3 had 1 children
[robot_state_publisher-1] Link link4 had 1 children
[robot_state_publisher-1] Link link5 had 0 children
[robot_state_publisher-1] [INFO] [1659336816.100317312] [robot_state_publisher]: got segment base
[robot_state_publisher-1] [INFO] [1659336816.100434612] [robot_state_publisher]: got segment link1
[robot_state_publisher-1] [INFO] [1659336816.100443491] [robot_state_publisher]: got segment link2
[robot_state_publisher-1] [INFO] [1659336816.100451008] [robot_state_publisher]: got segment link3
[robot_state_publisher-1] [INFO] [1659336816.100457862] [robot_state_publisher]: got segment link4
[robot_state_publisher-1] [INFO] [1659336816.100464597] [robot_state_publisher]: got segment link5
[rviz2-2] [INFO] [1659336816.529758865] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-2] [INFO] [1659336816.530082788] [rviz2]: OpenGL version: 3.1 (GLSL 1.4)
[rviz2-2] [INFO] [1659336816.620671305] [rviz2]: Stereo is NOT SUPPORTED
[follow_display-3] [INFO] [1659336816.635957580] [follow_display]: port:/dev/ttyUSB0, baud:115200
[rviz2-2] Parsing robot urdf xml string.
```

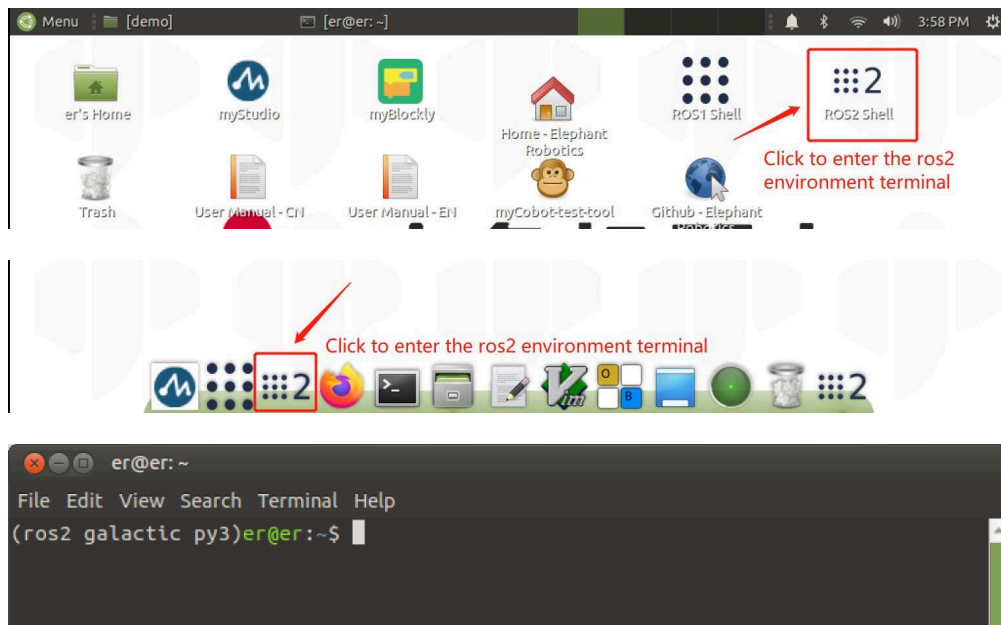
Then open another command line and run:

- mecharm 270-M5 version:

```
ros2 run mecharm teleop_keyboard
```

- mecharm 270-PI version:

Click the ROS2 Shell icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



Then run the command:

```
ros2 run mecharm_pi teleop_keyboard
```

You will see the following output in the command line:

```
mecharm Teleop Keyboard Controller
-----
Moving options(control coordinations [x,y,z,rx,ry,rz]):
    w(x+)

    a(y-)    s(x-)    d(y+)

    z(z-)    x(z+)

u(rx+)    i(ry+)    o(rz+)
j(rx-)    k(ry-)    l(rz-)

Gripper control:
    g - open
    h - close

Other:
    1 - Go to init pose
    2 - Go to home pose
    3 - Resave home pose
    q - Quit

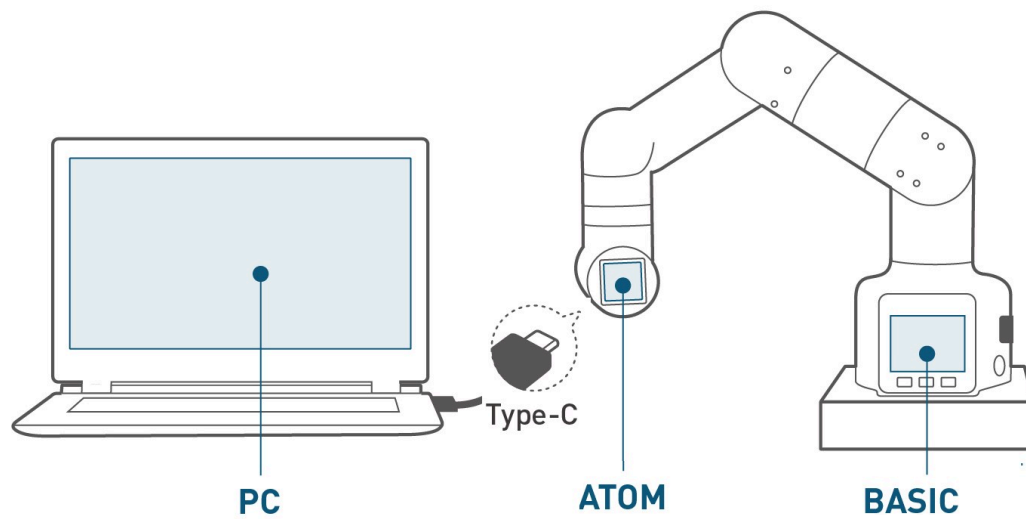
currently:    speed: 40    change percent: 5
```

Parameters supported by this script:

- `_speed`: the movement speed of the robot arm
- `_change_percent`: movement distance percentage

Communication and Message Commands

Notice: Before the direct communication via communication protocol, **Transponder** is required to be burnt on M5Stack-basic, and the latest atomMain on Atom.



1 Mechanical arm motion parameters

Joint	minimum value°	maximum value°	Maximum speed (°/s)	Maximum acceleration (°/s ²)
J1	-168	168	150	200
J2	-135	135	150	200
J3	-150	150	150	200
J4	-145	145	150	200
J5	-165	165	150	200
J6	-180	180	150	200

Axis	minimum value /mm	maximum value/mm**	Maximum speed (mm/s)	Maximum acceleration(mm/s ²)
x	-281.45	281.45	100	400
y	-281.45	281.45	100	400
z	-70	412.76	100	400
rx	-180°	180°	40°	66°/s ²
ry	-180°	180°	40°	66°/s ²
rz	-180°	180°	40°	66°/s ²

2 Settings of USB Communication

Make sure the following settings are prepared:

- Interface of mainline: USB Type-C connection
- Port ratio: 11520
- Data bit: 8
- Parity bit: none
- Stop bit: 1

3 Introduction to Command Frame & Sole Instruction

The main PC transmit data via M5Stakc-basic to peripheral PC. The peripheral PC decodes the data like commands with return values and then send the results back to the main PC within 500ms.

4 Formats of Message Commands' Sending and Receiving

Both sending and receiving should be represented in hexadecimal. Each command should contain 5 parts as shown below. Part 3 and 4 can be left a blank.

- 1 Pin of command: 0xFE 0xFE
 - Invariable
 - Indispensable
- 2 Effective lengthen:
 - Aggregated length including pin, serial number, functional codes and end
 - Indispensable
- 3 Serial number: 00 ~ 8F
 - Corresponding number of developed commands
 - You may leave it blank.
- 4 Functional codes:
 - Purpose-oriented
 - You may leave it blank.
- 5 End: 0XFA
 - Invariable
 - Indispensable

5 Explanation for Commands

The main PC transmit data via M5Stakc-basic to peripheral PC. The peripheral PC decodes the data like commands with return values and then send the results back to the main PC within 500ms.

Type	Data	Length	Function
Command Frame	start bit: 0	1	Head frame identification, 0XFE
	start bit: 1	1	Head frame identification, 0XFE
	bit of data length	1	Different commands correspond to different data length
	command bit	1	depending on different commands
Command Frame	data	0-16	commands with data, depending on different commands
End Frame	end bit	1	stop bit, 0XFA

6 Explanation for Sole-Instruction Commands

Powering up

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X10
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 10 FA

no return value

Power Decreasing and Connection Breaking up

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X11
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 11 FA

no return value

Checking the Status of Atom

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X12
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 12 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X12
Data[4]	power on/off	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that Atom is powered on:

port return: FE FE 03 12 01 FA

Power Decreasing

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X13
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 13 FA

no return value

Robot System Checking: Normal

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X14
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 14 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X14
Data[4]	connection/connection breaking up	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that Atom is successfully connected:

port return: FE FE 03 14 01 FA

Command Updating Mode (Interpolation Setting/Motion Mode Updating)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X16
Data[4]	connection/connection breaking up	0X01/0X00
Data[5]	end frame	0XFA

Example:

1. Setting updating motion mode:

Port transmission: FE FE 03 16 01 FA

2. Setting interpolation motion mode:

Port transmission: FE FE 03 16 00 FA

Free Mode (switch off torsion output)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X1A
Data[4]	turn on/off	01/00
Data[5]	end frame	0XFA

Example:

Setting free motion mode:

Port transmission: FE FE 03 1A 01 FA

Checking whether free mode is set

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X1B
Data[5]	end frame	0XFA

Example:

Port return: FE FE 02 1B FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	command frame	0X1B
Data[4]	turn on/off	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that Atom is in free mode:

port return: FE FE 03 1B 01 FA

Reading Angles (blocking information)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X20
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 20 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X0E
Data[3]	return value: command frame	0X20
Data[4]	high angle of No.1 steering gear	Angle1_high
Data[5]	low angle of No.1 steering gear	Angle1_low
Data[6]	high angle of No.2 steering gear	Angle2_high
Data[7]	low angle of No.2 steering gear	Angle2_low
Data[8]	high angle of No.3 steering gear	Angle3_high
Data[9]	low angle of No.3 steering gear	Angle3_low
Data[10]	high angle of No.4 steering gear	Angle4_high
Data[11]	low angle of No.4 steering gear	Angle4_low
Data[12]	high angle of No.5 steering gear	Angle5_high
Data[13]	low angle of No.5 steering gear	Angle5_low
Data[14]	high angle of No.6 steering gear	Angle6_high
Data[15]	low angle of No.6 steering gear	Angle6_low
Data[16]	end frame	0XFA

Example:

Return value of port: FE FE 0E 20 00 8C 00 3D FF E6 FF 3F 00 AF FF 51 FA

How to get the angle of joint 1:

$temp = angle1_low + angle1_high * 256$

$Angle1 = (temp \setminus 33000 \ ?(temp - 65536) : temp) / 100$

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

Other joint angles are counted in a similar way.

Sending Sole Angle

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X21
Data[4]	serial number of steering gear	joint_no
Data[5]	high angle	angle_high
Data[6]	low angle	angle_low
Data[7]	specified speed	sp
Data[8]	end frame	0XFA

Example:

Let the No.1 steering gear move to zero position

Port transmission: FE FE 06 21 01 00 00 14 FA

joint number: 1-6

angle_high: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

angle_low: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

no return value

Sending Entire Angles

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0F
Data[3]	command frame	0X22
Data[4]	high angle of No.1 steering gear	Angle1_high
Data[5]	low angle of No.1 steering gear	Angle1_low
Data[6]	high angle of No.2 steering gear	Angle2_high
Data[7]	low angle of No.2 steering gear	Angle2_low
Data[8]	high angle of No.3 steering gear	Angle3_high
Data[9]	low angle of No.3 steering gear	Angle3_low
Data[10]	high angle of No.4 steering gear	Angle4_high
Data[11]	low angle of No.4 steering gear	Angle4_low
Data[12]	high angle of No.5 steering gear	Angle5_high
Data[13]	low angle of No.5 steering gear	Angle5_low
Data[14]	high angle of No.6 steering gear	Angle6_high
Data[15]	low angle of No.6 steering gear	Angle6_low
Data[16]	specified speed	Sp
Data[17]	end frame	0XFA

Example:

Send 0 angle to entire joint, and let the steering gear move to zero position.

Port transmission: FE FE 0F 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1E FA

angle1_high: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

angle1_low: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Other angles are counted in a similar way.

no return value

Reading the Entire Coordinates

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X23
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 23 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X0E
Data[3]	return value: command frame	0X23
Data[4]	specify high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify high status of z coordinate	z_high
Data[8]	specify low status of z coordinate	z_low
Data[9]	low angle of No.3 steering gear	Angle3_low
Data[10]	high angle of No.4 steering gear	Angle4_high
Data[11]	low angle of No.4 steering gear	Angle4_low
Data[12]	high angle of No.5 steering gear	Angle5_high
Data[13]	low angle of No.5 steering gear	Angle5_low
Data[14]	high angle of No.6 steering gear	Angle6_high
Data[15]	low angle of No.6 steering gear	Angle6_low
Data[16]	end frame	0XFA

Example:

Port return value: FE FE 0E 23 01 BC FD A0 10 15 DC 66 FF 54 DE 21 FA

How to get x coordinate:

$temp = x_low + x_high * 256$

$x \text{ coordinate} = (temp \setminus 33000 ? (temp - 65536) : temp) / 10$

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

y coordinate is counted in a similar way.

How to get rx coordinate:

$temp = rx_low + rx_high * 256$

$x \text{ coordinate} = (temp \setminus 33000 ? (temp - 65536) : temp) / 100$

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

y coordinate is counted in a similar way.

Sending Parameters of Sole Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X24
Data[4]	axis	x/y/z/rx/ry/rz
Data[5]	specify high status of the parameters of xyz/rxryrz	xyz/ rxryrz_high
Data[6]	specify low status of the parameters of xyz/rxryrz	xyz/rxryrz_low
Data[7]	specified speed	Sp
Data[8]	end frame	0XFA

Example:

Given that the x coordinate is 200 and target speed is 20,

port transmission: FE FE 06 24 01 07 D0 14 FA

specified axis coordinate: byte

range: 1-6

data type of xyz_high: byte

counting method: value of x/y/z coordinate is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of xyz_low: byte

counting method: value of x/y/z coordinate is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

data type of rxryrz_high: byte

counting method: value of rx/ry/rz coordinate is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of rxryrz_low: byte

counting method: value of rx/ry/rz coordinate is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

no Return Value

Sending Parameters of Entire Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X10
Data[3]	command frame	0X25
Data[4]	specify high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	specify speed	Sp
Data[17]	mode	0X01
Data[17]	end frame	0XFA

Example:

Given that the coordinate of the end of robotic arm is (150.3, -68.7, 101.8, 10.18, 0, -90) , and speed is 10,

port transmission: FE FE 10 25 05 DF FD 51 03 FA BC 30 00 00 DC D8 0A 01 FA

data type of x_high: byte

counting method: value of x coordinate is multiplied by 10 and get the hexadecimal highbyte

data type of x_low: byte

counting method: value of x coordinate is multiplied by 10 and get the hexadecimal highbyte

y coordinate is counted in a similar way.

data type of rx_high: byte

counting method: value of rx coordinate is multiplied by 10 and get the hexadecimal highbyte

data type of rx_low: byte

counting method: value of rx coordinate is multiplied by 100 and get the hexadecimal highbyte

ry coordinate is counted in a similar way.

no return value

Program pause

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X26
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 26 FA

no return value

Judging Whether Program Stops

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X27
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 27 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X27
Data[4]	suspend/not suspend	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that program suspends,

port return value: FE FE 03 27 01 FA

Program Recover

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X28
Data[4]	end frame	0XFA

Port transmission: FE FE 02 28 FA

no return value

Program Stops

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X29
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 29 FA

no return value

Whether Reaching Specified Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E/0X0F
Data[3]	command frame	0X2A
Data[4]	high status of x coordinate/highbyte of angle of No.1 steering gear	x_high/Angle1_high
Data[5]	low status of x coordinate/lowbyte of angle of No.1 steering gear	x_low/Angle1_low
Data[6]	high status of y coordinate/highbyte of angle of No.2 steering gear	y_high/Angle2_high
Data[7]	low status of y coordinate/lowbyte of angle of No.2 steering gear	y_low/Angle2_low
Data[8]	high status of y coordinate/highbyte of angle of No.3 steering gear	z_high/Angle3_high
Data[9]	low status of z coordinate/lowbyte of angle of No.3 steering gear	z_low/Angle3_low
Data[10]	high status of rx coordinate/highbyte of angle of No.4 steering gear	rx_high/Angle4_high
Data[11]	low status of rx coordinate/lowbyte of angle of No.4 steering gear	rx_low/Angle4_low
Data[12]	high status of ry coordinate/highbyte of angle of No.5 steering gear	ry_high/Angle5_high
Data[13]	low status of ry coordinate/lowbyte of angle of No.5 steering gear	ry_low/Angle5_low
Data[14]	high status of rz coordinate/highbyte of angle of No.6 steering gear	rz_high/Angle6_high
Data[15]	low status of rz coordinate/lowbyte of angle of No.6 steering gear	rz_low/Angle6_low
Data[16]	coordinate/angle	0X01/0X00
Data[17]	end frame	0XFA

Example:

Judging whether robotic arm move to zero position:

port transmission: FE FE 0F 2A 00 00 00 00 00 00 00 00 00 00 00 00 FA

data type of x_high: byte

counting method: value of x coordinate is multiplied by 10 and converted into integral form to get the hexadecimal highbyte

data type of x_low: byte

counting method: value of x coordinate is multiplied by 10 and converted into integral form to get the hexadecimal lowbyte

y coordinate is counted in a similar way.

data type of rx_high: byte

counting method: value of rx coordinate is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of rx_low: byte

counting method: value of rx coordinate is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

ry coordinate is counted in a similar way.

data type of angle_high: byte

counting method: value angle is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle_low: byte

counting method: value of angle is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Type: byte (temporarily unavailable)

Return Value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X2a
Data[4]	reaching the point/not reaching the point	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that the robotic arm doesn't reach the point:

port return value: FE FE 03 2A 00 FA

Movement Check

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X2B
Data[4]	end frame	0XFA

Example:

port transmission: FE FE 02 2B FA

Return Value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X2B
Data[4]	in motion/not in motion	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that the program is running:

port return value: FE FE 03 2B 01 FA

jog-Joint-Oriented Movement

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X30
Data[4]	serial number of steering gear	Joint
Data[5]	direction of steering gear	direction
Data[6]	specified speed	sp
Data[7]	end frame	0XFA

Example:

Given that NO.1 steering gear is revolving clockwise at the speed of 20%:

port return value: FE FE 05 30 01 01 14 FA

range of serial number of joint: 1-6

di: data type of byte, either 0 or 1

sp: data type of byte, ranging from 0 to 100

no return value.

jod-Absolute Control

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X31
Data[4]	serial number of steering gear	Joint
Data[5]	highbyte of angle of steering gear	Angle_high
Data[6]	lowbyte of angle of steering gear	Angle_low
Data[7]	specified speed	sp
Data[8]	end frame	0XFA

Example:

Given that No.1 steering gear moves to 45° at the speed of 20

port transmission: FE FE 06 31 01 11 94 14 FA

range of serial number of joint: 1-6

data type of Angle_high: byte

counting method: value of Angle_high is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of Angle_low: byte

counting method: value of Angle_low is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

sp: data type of byte, ranging from 0 to 100

no return value

jog-Coordinate-Oriented Movement

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X32
Data[4]	specified coordinate	axis
Data[5]	direction of steering gear	di
Data[6]	specified speed	sp
Data[7]	end frame	0XFA

Example:

Given that robotic arm moves towards x coordinate at the speed of 20

port transmission: FE FE 06 32 01 01 14 FA

axis ranges from 1 to 6, representing x, y, z, rx, ry, rz

di: data type of byte, either 0 or 1

sp: data type of byte, ranging from 0 to 100

no return value

jog-Stepper Model

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X31
Data[4]	serial number of steering gear	Joint
Data[5]	highbyte of angle of steering gear	Angle_high
Data[6]	lowbyte of angle of steering gear	Angle_low
Data[7]	specified speed	sp
Data[8]	end frame	0XFA

Example:

Given that the angle of No.1 steering expand by 45° and revolves at the speed of 20

port transmission: FE FE 06 33 01 11 94 14 FA

serial number of joint ranges from 1 to 6

data type of angle_high: byte

counting method: value angle is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle_low: byte

counting method: value of angle is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

sp: data type of byte, ranging from 0 to 100

no return value

Sending Potential

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X3A
Data[4]	serial number of steering gear	Joint
Data[5]	high status of potential	Encoder_high
Data[6]	low status of potential	Encoder_low
Data[7]	specified speed	sp
Data[8]	end frame	0XFA

Example:

Given that No.5 joint is set to 2048 and revolves at the speed of 20

port transmission: FE FE 05 3A 05 08 00 14 FA

serial number of joint ranges from 1 to 6

data type of Joint: byte

data type of Encoder_high: byte

counting method: get the high status of potential (in hexadecimal form)

data type of Encoder_low: byte

counting method: get the low status of potential (in hexadecimal form)

no return value

Get the Potential

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X3B
Data[4]	serial number of joint	joint
Data[5]	end frame	0XFA

Example:

get the potential of NO.2 steering gear

serial number of joint ranges from 1 to 6

Return Value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	return value: command frame	0X3B
Data[4]	high status of potential	Encoder_high
Data[5]	low status of potential	Encoders_low
Data[6]	end frame	0XFA

Example:

port transmission: FE FE 04 3B 08 07 FA

How potentials are counted:

potential = low status of potential + high status of potential * 256

Sending Potential of Six Steering Gears

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E/0X0F
Data[3]	command frame	0X3C
Data[4]	highbyte of No.1 steering gear	encoder_1_high
Data[5]	lowbyte of No.1 steering gear	encoder_1_low
Data[6]	highbyte of No.2 steering gear	encoder_2_high
Data[7]	lowbyte of No.2 steering gear	encoder_2_low
Data[8]	highbyte of No.3 steering gear	encoder_3_high
Data[9]	lowbyte of No.3 steering gear	encoder_3_low
Data[10]	highbyte of No.4 steering gear	encoder_4_high
Data[11]	lowbyte of No.4 steering gear	encoder_4_low
Data[12]	highbyte of No.5 steering gear	encoder_5_high
Data[13]	lowbyte of No.5 steering gear	encoder_5_low
Data[14]	highbyte of No.6 steering gear	encoder_6_high
Data[15]	lowbyte of No.6 steering gear	encoder_6_low
Data[16]	specified speed	Sp
Data[17]	end frame	0XFA

Example:

Given that the potential of all is 2048, and the speed is 20

port transmission: FE FE 0F 3C 08 00 08 00 08 00 08 00 08 00 14 FA

Refer to the chart above for each potential.

data type of encoder_1_high: byte

counting method: convert the potential of No.1 steering gear into integral form and get the highbyte hexadecimal

data type of encoder_1_low: byte

counting method: convert the potential of No.1 steering gear into integral form and get the lowbyte hexadecimal

sp: data type of byte, ranging from 0 to 100

no return value

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X3D
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 3D FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X3D
Data[4]	highbyte of No.1 steering gear	encoder_1_high
Data[5]	lowbyte of No.1 steering gear	encoder_1_low
Data[6]	highbyte of No.2 steering gear	encoder_2_high
Data[7]	lowbyte of No.2 steering gear	encoder_2_low
Data[8]	highbyte of No.3 steering gear	encoder_3_high
Data[9]	lowbyte of No.3 steering gear	encoder_3_low
Data[10]	highbyte of No.4 steering gear	encoder_4_high
Data[11]	lowbyte of No.4 steering gear	encoder_4_low
Data[12]	highbyte of No.5 steering gear	encoder_5_high
Data[13]	lowbyte of No.5 steering gear	encoder_5_low
Data[14]	highbyte of No.6 steering gear	encoder_6_high
Data[15]	lowbyte of No.6 steering gear	encoder_6_low
Data[17]	end frame	0XFA

Example:

Given that all robotic arms are set in zero position,

port return value: FE FE 0E 3D 08 00 08 00 08 00 08 00 08 00 08 00 FA

How to count potential

How potentials are counted:

potential = low status of potential + high status of potential * 256

Reading Potential of Six Steering Gears

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X3D
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 3D FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X3D
Data[4]	highbyte of No.1 steering gear	encoder_1_high
Data[5]	lowbyte of No.1 steering gear	encoder_1_low
Data[6]	highbyte of No.2 steering gear	encoder_2_high
Data[7]	lowbyte of No.2 steering gear	encoder_2_low
Data[8]	highbyte of No.3 steering gear	encoder_3_high
Data[9]	lowbyte of No.3 steering gear	encoder_3_low
Data[10]	highbyte of No.4 steering gear	encoder_4_high
Data[11]	lowbyte of No.4 steering gear	encoder_4_low
Data[12]	highbyte of No.5 steering gear	encoder_5_high
Data[13]	lowbyte of No.5 steering gear	encoder_5_low
Data[14]	highbyte of No.6 steering gear	encoder_6_high
Data[15]	lowbyte of No.6 steering gear	encoder_6_low
Data[17]	end frame	0XFA

Example:

Given that all robotic arms are set in zero position,

port return value: FE FE 0E 3D 08 00 08 00 08 00 08 00 08 00 08 00 FA

How potentials are counted:

potential = low status of potential + high status of potential * 256

Setting Speed

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X41
Data[4]	specified speed	Sp
Data[5]	end frame	0XFA

Sp: data type of byte, ranging from 0 to 100

Example:

Given that the speed is 50%

port return value: FE FE 03 41 32 FA

Reading the Smallest Angle

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X4A
Data[4]	serial number of steering gear	Joint_number
Data[5]	end frame	0XFA

Example:

Reading the smallest angle of joint 2

Port transmission: FE FE 03 4A 02 FA

serial number of joint ranges from 1 to 6

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	return value: command frame	0X4A
Data[4]	serial number of joint	Joint_number
Data[5]	high angle of steering gear	Angle_high
Data[6]	low angle of steering gear	Angle_low
Data[7]	end frame	0XFA

Example:

port return value: FE FE 05 4A 02 F9 F2 FA

How smallest angles are counted:

temp = angle1_low+angle1_high*256

Angle1= (temp \ 33000 ?(temp – 65536) : temp) /10

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 10; if temp is less than 33000, then temp is divided by 10 directly.

Reading the Smallest Angle

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X4A
Data[4]	serial number of steering gear	Joint_number
Data[5]	end frame	0XFA

Serial number of joint ranges from 1 to 6

Example:

Reading the largest angle of joint 2

Port transmission: FE FE 03 4B 02 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X05
Data[3]	return value: command frame	0X4A
Data[4]	serial number of joint	Joint_number
Data[5]	high angle of steering gear	Angle_high
Data[6]	low angle of steering gear	Angle_low
Data[7]	end frame	0XFA

Example:

port return value: FE FE 05 4B 02 06 72 FA

How the largest angles are counted:

temp = angle1_low+angle1_high*256

Angle1= (temp \ 33000 ?(temp – 65536) : temp) /10

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 10; if temp is less than 33000, then temp is divided by 10 directly.

Setting the Smallest Angle

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X05
Data[3]	return value: command frame	0X4C
Data[4]	serial number of joint	Joint_number
Data[5]	high angle of steering gear	Angle_high
Data[6]	low angle of steering gear	Angle_low
Data[7]	end frame	0XFA

Example:

Given that the smallest angle of joint 2 is 0

Serial number of joint ranges from 1 to 6

data type of angle1_high: byte

counting method: angle is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle1_low: byte

counting method: angle is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Port transmission: FE FE 05 4C 02 00 00 FA

no return value

Setting the Largest Angle

Domain	Explanation	Data
Data[0]	identification frame	0xFE
Data[1]	identification frame	0xFE
Data[2]	data-length frame	0x05
Data[3]	command frame	0x4D
Data[4]	serial number of joint	Joint_number
Data[5]	highbyte of steering gear	Angle_high
Data[6]	lowbyte of steering gear	Angle_low
Data[7]	end frame	0xFA

Example:

Given that the largest angle of joint 2 is 45

Serial number of joint ranges from 1 to 6

data type of angle1_high: byte

counting method: angle is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle1_low: byte

counting method: angle is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Port transmission: FE FE 05 4C 02 11 94 FA

no return value

Checking Connection

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X50
Data[4]	serial number of steering gear	Joint_number
Data[5]	end frame	0XFA

Serial number of joint ranges from 1 to 6

Example:

Checking whether No.1 steering gear is connected

Port transmission: FE FE 03 50 01 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	command frame	0X50
Data[4]	serial number of joint	Joint_number
Data[5]	connected/not connected	0X01/0X00
Data[6]	end frame	0XFA

Example:

No.1 steering gear is connected

Port return value: FE FE 04 50 01 01 FA

Checking Whether All Steering Gears Are Powered On

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X51
Data[5]	end frame	0XFA

Example:

Port transmission: FE FE 02 51 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X51
Data[4]	power on/off	0X01/0X00
Data[5]	end frame	0XFA

Example:

Not all steering gears are powered on

Port transmission: FE FE 03 51 01 FA

Reading Servo Parameters

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X53
Data[4]	serial number of joint	Joint_number
Data[5]	data address	data_id
Data[6]	end frame	0XFA

Example:

Reading ratio parameter of P position of No.1 steering gear

Port transmission: FE FE 03 51 01 FA

Serial number of joint ranges from 1 to 6

Data_id: data type byte, referring to the chart below for specific value:

Address	Function	Range	Initial Value	Explanation
20	LED siren	0-254	0	1\0: turn on/off LED siren
21	position loop P	0-254	10	control the ratio coefficient
22	position loop I	0-254	0	control the differential coefficient
23	position loop D	0-254	1	control the integral-action coefficient
24	minimum starting force	0-1000	0	set the smallest torque output capability 1000 = 100%

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X53
Data[4]	return value: data	data
Data[5]	end frame	0XFA

Setting Servo Parameters

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X52
Data[4]	serial number of joint	Joint_no
Data[5]	data address	data_id
Data[6]	end frame	0XFA

Example:

Reading ratio parameter of P position of No.1 steering gear

Port transmission: FE FE 05 52 01 15 01 FA

Serial number of joint ranges from 1 to 6

no return value

Data_id: data type byte, referring to the chart below for specific value:

Address	Function	Range	Initial Value	Explanation
20	LED siren	0-254	0	1\0: turn on/off LED siren
21	position loop P	0-254	10	control the ratio coefficient
22	position loop I	0-254	0	control the differential coefficient
23	position loop D	0-254	1	control the integral-action coefficient
24	minimum starting force	0-1000	0	set the smallest torque output capability 1000 = 100%

Setting Zero Position of Steering Gear

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X54
Data[4]	serial number of joint	Joint_no
Data[6]	end frame	0XFA

Example:

Reading zero position of No.1 steering gear

Port transmission: FE FE 03 54 01 FA

Serial number of joint ranges from 1 to 6

no return value

Braking Single Motor

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X55
Data[4]	serial number of joint	Joint_no
Data[5]	end frame	0XFA

Example:

Serial number of joint ranges from 1 to 6

Braking No.1 steering gear,

Port transmission: FE FE 03 55 01 FA

no return value

Power Failure of Single Motor

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X56
Data[4]	serial number of steering gear	Servo_no
Data[5]	end frame	0XFA

Example:

Let No.3 steering gear reducing power

Serial number of servo ranges from 1 to 6

no return value

Powering On a Single Motor

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X57
Data[4]	serial number of steering gear	Servo_no
Data[5]	end frame	0XFA

Example:

power on No.1 steering gear

Port transmission: FE FE 03 57 01 FA

Serial number of servo ranges from 1 to 6

no return value

Setting Mode of Atom

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X60
Data[4]	serial number of pin	pin_no
Data[5]	input/output	00X00/00X01
Data[6]	end frame	0XFA

Example:

setting atom pin22 as input mode

Port transmission: FE FE 04 60 16 00 FA

Pin_no: data type byte

Pin_mode: 0/1

no return value

Setting Atom IO (setDigitalOutput)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X61
Data[4]	serial number of pin	pin_no
Data[5]	signal of electrical level	00X00/00X01
Data[6]	end frame	0XFA

Example:

setting atom pin23 as high electrical level

Port transmission: FE FE 04 61 17 01 FA

no return value

Reading Atom IO(getDigitalInput)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X61
Data[4]	serial number of pin	pin_no
Data[5]	end frame	0XFA

Example:

reading signal of electrical level of pin23

Port transmission: FE FE 03 62 16 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X53
Data[4]	pin number	pin_no
Data[5]	signal of electrical level	0X00/0X01
Data[6]	end frame	0XFA

Example:

Given that pin 22 is of high electrical level

Port return value: FE FE 04 62 16 01 FA

Reading the Angle of Gripper

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X65
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 65 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X65
Data[4]	span of gripper's opening	value
Data[5]	end frame	0XFA

value: 0-100%

Example:

Given that gripper is completely opening

Port return value: FE FE 03 65 64 FA

value: $6 * 16 + 4 = 100$

Setting the Mode of Gripper

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X66
Data[4]	gripper opening/closing	0X00/0X01
Data[5]	speed	Sp
Data[6]	end frame	0XFA

Example:

making gripper open at the speed of 50

Port transmission: FE FE 04 66 00 32 FA

no return value

Setting Angles of Gripper

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X67
Data[4]	span of gripper's opening/closing	value
Data[5]	speed	Sp
Data[6]	end frame	0XFA

Example:

Given that gripper opens by 50% at the speed of 30

Port transmission: FE FE 04 67 32 14 FA

The value can be converted into hexadecimal

no return value

Setting the Gripper to Zero Position

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X68
Data[4]	end frame	0XFA

Example:

Setting the Gripper to Zero Position

Port transmission: FE FE 02 68 FA

Checking Whether the Gripper is Moving

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X69
Data[4]	end frame	0XFA

Port transmission: FE FE 02 69 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X69
Data[4]	stop/move	00/01
Data[6]	end frame	0XFA

Given that gripper stops working

Port transmission: FE FE 03 69 00 FA

Setting RGB of Atom

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X6A
Data[4]	R	0X00/0XFF
Data[5]	G	0X00/0XFF
Data[6]	B	0X00/0XFF
Data[7]	end frame	0XFA

Example:

Setting RGB color as blue

Port transmission: FE FE 05 6A 00 00 FF FA

no return value

Setting IO Output

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0Xa0
Data[4]	pin number	Pin_no
Data[5]	signal of electrical level	0X00/0X01
Data[4]	end frame	0XFA

Example:

Setting high electrical level of pin 2

Port transmission: FE FE 02 a0 02 01 FA

Reading IO Output

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0Xa1
Data[4]	pin number	Pin_no
Data[4]	end frame	0XFA

Port transmission: FE FE 02 a0 02 01 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	return value: command frame	0Xa1
Data[4]	pin number	Pin_no
Data[5]	signal of electrical level	0X00/0X01
Data[6]	end frame	0XFA

Example:

Given that pin 2 has high electrical level

Port return value: FE FE 04 a1 02 01 FA

Acquirng WiFi Account & Password

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0Xb1
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 b1 FA

port return value: ssid: MyCobotWiFi2.4G password: mycobot123

ssid: WiFi account

password: WiFi password

Setting Port Number

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0Xb2
Data[4]	highbyte of port	port_high
Data[5]	lowbyte of port	port_low
Data[6]	end frame	0XFA

Example:

Given that port number is 7000

Port transmission: FE FE 04 b2 1b 58 FA

port_high: highbyte of hexadecimal port number

port_low: lowbyte of hexadecimal port number

No return value

Setting Tool Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X81
Data[4]	specify high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	end frame	0XFA

Example:

Given that (0, 0, 50, 0, 0, 0) is tool coordinate,

Port transmission: FE FE 0E 81 00 00 00 00 13 88 00 00 00 00 00 00 FA

no return value

Acquiring Tool Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X82
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 04 b2 1b 58 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X81
Data[4]	specify high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	end frame	0XFA

Port return value: FE FE 0E 82 00 00 00 00 13 88 00 00 00 00 00 00 FA

How to get x coordinate:

$\text{temp} = x_low + x_high * 256$

$x \text{ coordinate} = (\text{temp} \setminus 33000 \ ? (\text{temp} - 65536) : \text{temp}) / 10$

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

y coordinate is counted in a similar way.

How to get rx coordinate:

$\text{temp} = rx_low + rx_high * 256$

$x \text{ coordinate} = (\text{temp} \setminus 33000 \ ? (\text{temp} - 65536) : \text{temp}) / 100$

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

ry coordinate is counted in a similar way.

Setting World Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X81
Data[4]	specify high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	end frame	0XFA

Port return value: FE FE 0E 84 00 00 00 00 13 88 00 00 00 00 00 00 FA

How to get x coordinate:

$\text{temp} = x_low + x_high * 256$

$x \text{ coordinate} = (\text{temp} \setminus 33000 \ ? (\text{temp} - 65536) : \text{temp}) / 10$

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

y coordinate is counted in a similar way.

How to get rx coordinate:

$\text{temp} = rx_low + rx_high * 256$

$x \text{ coordinate} = (\text{temp} \setminus 33000 \ ? (\text{temp} - 65536) : \text{temp}) / 100$

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

ry coordinate is counted in a similar way.

Setting Base Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X85
Data[4]	base coordinate/world coordiante	00/01
Data[5]	end frame	0XFA

Example:

Given that the coordinate should be set as world coordinate

port transmission: FE FE 03 85 01 FA

no return value

Acquiring Base Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X86
Data[4]	end frame	0XFA

Example:

port transmission: FE FE 02 86 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X86
Data[4]	base coordinate/world coordinate	00/01
Data[5]	end frame	0XFA

Example:

port transmission: FE FE 03 86 01 FA

Setting End Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X89
Data[4]	flange	00/01
Data[5]	end frame	0XFA

Example:

Given that end coordinate are set as tool

port transmission: FE FE 03 89 01 FA

no return value

Acquiring End Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X8a
Data[4]	end frame	0XFA

Example:

port transmission: FE FE 02 8a FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X8a
Data[4]	flange	00/01
Data[5]	end frame	0XFA

Example:

port return value: FE FE 03 86 01 FA

Appendix

The specific method of adding coordinate-exchanging programs from Atom repository and sports repository are listed as following.

1. Change end coordinate.
2. Set end coordinate by setEndType and getEndType.

FLANGE: Setting EndType as FLANG; TOOL: Setting EndType as TOOL End.

3. Read tool coordinate via setToolReference and getToolReference. (FLANGE serves as relative coordinate and information about tool end is relevant to FLANGE coordinate.)
4. After setting EndType as FLANGE, GetCoords and WriteCoords are counted depending on FLANG's position.
5. After setting EndType as TOOL, GetCoords and WriteCoords are counted depending on end's position.

6. Change base coordinate.
 7. Set base coordinate by setReferenceFrame.
-

RFTYPE: Base means setting robotic pedestal as base coordinate; RFTYPE:WORLD means make world coordinate as base coordinate; getReferenceFrame serves to read the type of base coordinate.

8. setWorldReference and getWorldReference works to read information about base coordinate. World coordinate serves as relative coordinate to type pedestal information of robots that is relevant to world coordinate.
9. If base coordinate acts as pedestal, GetCoords and WriteCoords take base coordinate as reference coordinate.
10. If world coordinate acts as pedestal, GetCoords and WriteCoords take world coordinate as reference coordinate.

Information Updating About Communication

These are newly-added functions: setting and reading of end coordinate, world coordinate, present coordinate, type of end, as well as moving method, and sending and receiving of information on robotic arms.

The communication is temporarily set as from 0x80 to 0x8A.

The roboticMessages space is specially added in ParameterList.h for more communication information. Now only `No Analytical Solution` signal has been added, while more will be added later.

The outline of MOVEJ are listed as follows:

First, count the Euclidean Distance between starting point and target point. Second, interpolation points are inserted every 10mm, based on the Euclidean Distance. If interpolation points do not have analytical solution, then search for plus or minus $\pi/30$ from three unchangeable direction for analytical solution. Avoid singular value and some distinctive position that cannot get analytical solution.

Change the interval between MOVEJ and JOG into dynamic time. And then, count the time spend moving through the maximum joint. And then, minus the moving time by specific period to take it as interval.

Message Board

💖 We're delighted to receive your comments, if you have any suggestions or questions about our gitbook, you can write here, we'll reply you as soon as possible, thanks ❤️

⚠️ If you can't see the message leaving place, it may be due to 🌐 network loading problems, please refresh the page, thanks !

